Subpixel image analysis

Gavin, John

Award date:
1995

Awarding institution:
University of Bath

Subpixel Image Analysis

submitted by

John Gavin

for the degree of Ph.D.

of the

University of Bath

1995

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

If a man will begin in certainties
he shall end in doubts;
but if he will be content to begin in doubts
he shall end in certainties.
Francis Bacon

We shall not cease from exploration
And the end of our exploring
Will be to arrive where we started
And know the place for the first time.
Thomas Stearns Eliot

Motivation

Image analysis is simply the science of extracting information from pictures. If information is encapsulated in an image, the human eye is usually very good at extracting qualitative information from the picture. Very often though, more consistent and objective methods are needed.

This thesis considers one specific application in image analysis, the detection and visualisation of edges around any objects in the image. Some original approaches to edge detection are developed and demonstrated. Efficient computational algorithms are defined and various applications are considered. The proposed techniques could be seen as a contribution to the field of image filtering.

Visualisation

Our primary objective is to improve the quality of the restoration at a boundary, between two regions of different colour, by allowing each pixel or voxel in an image to contain two colours.

The structure in the data is qualitatively presented by visualising the reconstruction of the unknown true image. We use a variety of presentation methods to visualise edges, with a precision that exceeds that of the underlying data. A concise but precise method of visualisation is very desirable to speed presentation, extract summary information and subsequently manipulate the data. This is particularly important in 3D where it is often necessary to deal with large volumes of data. Quantitative comparisions are also made.

Tools

Stochastic simulation models and Bayesian image analysis have provided a general framework to model image data and, in particular, to incorporate prior information. We use

a Bayesian model, which includes some prior knowledge, to reconstruct an image to a greater accuracy than that of the recording sensor. This prior knowledge, based on the edge length or surface area, is expressed through a prior probability for each image in a predefined class. The objective is to strike a balance between the discrete, noisy and blurred data and the class of possible reconstructions.

The models are implemented using algorithms based on Markov chain Monte Carlo theory. So we structure all algorithms to be *strictly local* in nature. This simplifies algorithms making them more transparent and thus easier to understand; but they are computationally intensive tools.

We discuss the design of algorithms for effecient exploration of two and three dimensional image spaces.

Applications

This thesis is primarily concerned with practical aspects of image analysis. Therefore to demonstrate the utility of the new algorithms, real datasets are used to address practical problems. The data and their associated problems dictate the design of the statistical models and the computational algorithms.

There are two primary datasets: one is two dimensional and the other is three dimensional. Their principal feature is that the size of the objects in the images is near the resolution of the recording device. Details are given about how the data was gathered and the scientists' objectives. The results of applying the algorithms to some simulated datasets are also presented.

Layout

The thesis is broken into the following parts:

- Chapter 1 provides some background to the subject of image analysis, leading to the reasons why subpixel analysis is important.
- Chapter 2 defines some notation and explains the statistical theory underlying the models that are discussed later.
- Chapter 3 considers a discrete, two-dimensional, subpixel image model.
- Chapter 4 discusses another two-dimensional subpixel image model, but assumes the true underlying image is continuous.
- Chapter 5 extends the model in Chapter 4 to three-dimensions.
- Finally, the conclusions are drawn and possible future work in outlined.

Miscellaneous issues are deferred to the appendices. A bibliography and index are included for reference.

Keywords

- Bayesian inference
- Conditional distributions
- Confocal microscopy
- Deconvolution
- Edge detection
- Gibbs sampler
- Hastings algorithm
- Image analysis
- Markov chain Monte Carlo
- Markov random fields
- Metropolis algorithm
- Spatial statistics
- Subpixel resolution
- Three-dimensional reconstruction

Contents

List of Figures

List of Tables

Abbreviations and symbols

Make everything as simple as possible but no simpler
Albert Einstein

The beginning of wisdom is the definition of terms.
Socrates

The page numbers in this glossary refer to the definition of each expression or its first occurrence in the text. In our notation, we sometimes do not distinguish between random variables, vectors and their values but we identify random variables by uppercase letters and vectors by bold type, when necessary.

Abbreviations and acronyms

2D ..... two dimensions 1
3D ..... three dimensions 66
CLSM ..... confocal laser scanning microscope 99
CMC ..... closest mean classifier 34
iff ..... if and only if 25
ICM ..... iterated conditional modes 37
g.c.d. ..... greatest common divisor 25
LOG ..... Laplacian-of-Gaussian filter 6
mb ..... megabytes 94
MAP ..... maximum a posteriori 38
MCMC ..... Markov chain Monte Carlo 4
MLE ..... maximum likelihood estimate 34
MPM ..... maximum marginal modes 39
MRF ..... Markov random field 21
p.d.f. ..... probability density function 22
SNR ..... signal-to-noise ratio 52

Symbols

$\cdot$ ..... dot product of two vectors 130
$\ll$ ..... a *lot* less than 80
$\stackrel{\text{def}}{=}$ ..... is defined to be 20
$\sim$ ..... a neighbourhood relation 21
$\square$ ..... end of proof or definition or remark or example 1
$\times$ ..... size and dimension of image e.g. 5×4 image 5
$\otimes$ ..... cross product of two vectors 130
α ..... the acceptance probability for a proposal 27
β ..... smoothing penalty in the energy function 40
β_i ..... smoothing penalty for the i^{th} order neighbourhood 48
$\beta_i^{(m)}$ ..... the i^{th} order interaction term in the prior model 51
δ_i ..... the set of neighbours of pixel i 22
δ_i^1 ..... the first order set of neighbours of pixel i 22
δ_i^2 ..... the second order set of neighbours of pixel i 22
Ω ..... the set of allowable reconstructions 20
$\Omega^{(m)}$ ..... set of reconstructions at the m^{th} level cascade 49
π ..... an ergodic distribution 26
$\pi^{(T)}$ ..... an ergodic distribution at temperature T 34
σ^2 ..... the noise in the data 24
$\|\mathbf{A}\|$ ..... $\sqrt{A_1^2 + \cdots + A_n^2}$ = norm of a vector 24
|AB| ..... distance between points A and B 80
$b(z_i, z)$ ..... blurring coefficient between z_i and $z \in \Re$ 23
b ..... the value of the foreground colour (black) 20
c ..... a clique 22
C ..... the set of all cliques 22
D_i ..... the period of state i 25
E ..... the edges in a graph G 154
$f_X(x)$ ..... pdf of $X = (X_1, X_2, \ldots, X_n)^T \in \Omega$ (prior dist'n) 20
$f_{X_i}(x_i)$ ..... the marginal density of pixel i 20
$f_{Y|X}(y, x)$ ..... the pdf of the observed data Y given $X = x$ 21
$f_{X|Y}(x, y)$ ..... the posterior density of X given Y 21
$f_Y(y)$ ..... the marginal density of Y 21
g ..... output from a linear filter 5
G ..... a graph consisting of nodes V and edges E 154
$h(x_i)$ ..... the colour of pixel i 23
I ..... an indicator function 40
$L(x)$ ..... the total edge length in the image x 67
$L_i(x)$ ..... the total edge length in pixel i 68
$L^{(m)}$ ..... the m^{th} level of the cascade 49
n ..... the number of pixels in an image 20

Symbols (cont'd)

$\mathcal{N}$ ..... the set of natural numbers 25
m ....index of the levels in a cascade where $m = 1, 2, \ldots, M$ 48
M .....the maximum number of levels in a cascade 48
p ..... window size for a filter 5
p_i ..... point in 3D space 130
P .....the proposal matrix for a Markov chain 27
$q(x, x')$ .....one-step transition probability 25
r .....the number of vertices in a Stage 4 reconstruction 79
Q ..... the transition matrix for a Markov chain 25
$S(x)$ .....the total surface area in the image x 105
U_X .....the energy function for the prior 22
$U_{X|Y}$ .....the energy function for the posterior 24
$\mathbf{v_i}$ .....vector in 3D space 130
V ..... the nodes in a graph G 154
w ..... the value of the background colour (white) 20
W ..... weights in a filter 5
$\bar{x}$ .....the empirical average of x 38
$x^{(t)}$ ..... state of the Markov chain at time $t = 0, 1, \ldots$ 38
$x(z)$ .....the colour of the image x at the point $z \in \Re^2$ 67
X ..... the true image 20
$X^{(m)}$ ..... the true image at the m^{th} cascade level 49
Y .....the observed data or record 21

Chapter 1

From image analysis to subpixel edges

Things are always at their best
in the beginning.
Blaise Pascal
Lettres Provinciales

There is no abstract art.
You must always start with something.
Afterwards you can remove all traces of reality.
Pablo Picasso

1.1 Broad view of image analysis

1.1.1 Motivation for studying image analysis

It is becoming increasingly popular in the applied sciences to convey information in the format of digital images. For some two dimensional images, the data are collected by a perspective projection of the objects of interest onto a planar grid of rectangular or square picture elements, called *pixels*. The light from these objects is focussed by a lens onto the recording sensor, known as a charge-coupled-device detector (CCD). The CCD then digitises the radiated intensity of the light source so that the data are grey-levels or a finite set of colours. In other cases, the two dimensional image may be a cut surface that transects a three dimensional (3D) object. Measurements on individual features, or on the image as a whole, must then be extracted and interpreted to obtain useful information from the objects.

Example 1.1. EXAMPLES OF IMAGES.
Figure 1-1 shows two examples of images that are discussed in greater detail later in this thesis. In Plot (a), the two dimensional (2D) image is of a fungus mycelium growing on a microscope slide. Scientists are interested in the feeding behaviour of the fungus, which is reflected in its spatial distribution across the slide and the length of its arms (see §4.4.2 on page 87). Plot (b) shows a slice through a three dimensional image of a single, plant cell. In this case, the feature of interest is the change in the volume and surface area of an individual cell, under different circumstances (see §5.1.1 on page 99). □

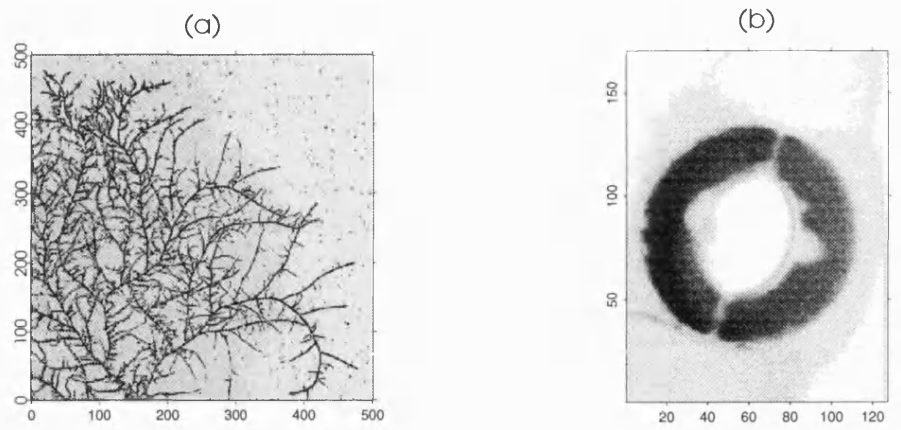

Figure 1-1: EXAMPLES OF IMAGES. In Plot (a), the image is of a fungus mycelium growing on a microscope slide. Scientists are interested in the feeding behaviour of the fungus, which is reflected in its spatial distribution across the slide and the length of its arms (see §4.4.2 on page 87). Plot (b) shows a slice through a 3D image of a plant cell, called a *guard cell*. When the cell contracts, the circular opening in the middle disappears. In this case, the feature of interest is the change in the volume and surface area of the cell between its opening and closing position (see §5.1.1 on page 99).

1.1.2 Subdividing image analysis

Level	Description
M+3	3-D scene interpretation
M+2	3-D scene description
M+1	2-D image description
6 to M	Higher level aggregation and model matching
5	Discovery of structural relationships
4	Feature classification or pattern recognition
3	Image segmentation and feature detection
2	Preprocessing and restoration
1	Sensor representation
0	Scene

Table 1.1: VARIOUS LEVELS OF ANALYSIS IN COMPUTER VISION. Levels 0 to 3 are called *image processing* or *low-level vision*. Higher levels are called *image interpretation* or *high-level vision*.

Image analysis and computer vision is a rapidly growing area. For the year 1991, Rosenfeld (1992) states that a selected bibliography of journal and conference proceedings for image analysis and computer vision runs to nearly 1,200 citations. The corresponding figure for the following year is 1,900 (Rosenfeld 1993).

Levine (1985) classifies the field into various levels, shown in Table 1.1. Rosenfeld's survey subdivides the field even further into topics like: applications, computational tech-

niques, feature detection, segmentation and three dimensional recovery-analysis, to name but a few. For example, the main results from Chapter 4, as presented in Gavin and Jennison (1995), fall into the category of 'feature detection' (Level 3), where the feature is the edges in the image, and into the sub-category 'computational techniques'.

Overall, the thesis deals with issues that fall into categories 0–3 and M+2.

1.1.3 Applications of image analysis

Image analysis and measurement methods are used in a broad range of applications in the applied sciences:

- satellite weather maps
- industrial quality control of macroscopic manufactured items
- light and electron microscopy of material structures
- biological, geological, astronomical or archaeological specimens
- integrated circuits, *etc.*

Generally, these methods are concerned with extracting a few numerical values, such as the number, size, shape or location of objects from the image. This may require image processing to correct defects, enhance some aspect of the image, compare multiple images, recognise objects of interest in a complex environment, or other steps. Ultimately, the image is reduced to just the features of interest, which may then require further editing, for instance to separate touching objects.

1.1.4 Reasons why statistical models are used in image analysis

Image analysis is simply the science of extracting information from pictures. The human eye is very good at extracting qualitative information but more consistent, objective methods are needed. Also automatic processing of images is becoming more essential, as more and more data are captured via images. Statistical methods can extract quantitative information automatically using computer based algorithms. We often want to count the number of objects in an image, estimate their areas, measure distances between objects, describe the shape of objects or find their boundaries. So in this section, we offer three distinct but related reasons *why* statistical science should be considered when processing and analysing digital images (Green 1994b; Green 1995a). In later chapters, we consider *how* this is done.

Use of probabilistic models for the true image

In statistical image models, the number of random variables is often of the same order as the number of data points. For example, each pixel in an image is often modelled as an individual random variable. So point estimates of the image features

of interest are difficult to find even after a large amount of computation. If we also want to estimate variability, standard numerical methods are usually inadequate.

A key feature of image analysis is that we often have prior information about the data. Consequently, a Bayesian framework is a natural way to incorporate the additional information, in the form of a prior distribution, in order to draw inferences about the image (Besag 1989). The prior distribution adds some stability and regularity to the model. It allows information about the image, such as local contextual regularities, to be approximately modelled using a Markov random field (see §2.2.3 on page 21).

This Bayesian formulation offers a unified approach to image analysis covering low level analysis, such as the removal of noise and blur, higher level work such as tomographic reconstruction, segmentation, texture modelling and, at the highest level, object recognition.

Modelling the degradation in the data

The degradation arises from transmission loss between the true image and the recorded data, the *record*. The loss of information is due to imperfections in the lens and the sensor, arising from blurring and noise. *Noise* is a disturbance in data that is either uninterpretable or not of interest. *Blurring* occurs when information, in the form of varying colours, 'leaks' into a pixel from its neighbouring pixels. A flexible likelihood is needed to reflect the characteristics of different sensing devices. Removing the degradation to draw inferences about the unknown true image, given the record, is a key objective of statistical image analysis (see §2.2.4 on page 23).

Designing efficient algorithms for stochastic simulation

Sophisticated optimisation algorithms are necessary, since we typically have to estimate a large number of parameters simultaneously. In some optimisation problems, the change in the cost function can be computed in an order of magnitude faster than the cost function itself. The key to such algorithms is the conditional independence between pixels, the random variables, given the parameters. Markov chain Monte Carlo (MCMC) methods provide the necessary tools to implement such algorithms.

We use the Bayesian method to produce point estimates of pixel images or image functionals using stochastic or deterministic algorithms. These algorithms are used to simulate both the prior and posterior distributions (see §2.3 on page 25). The algorithms are sets of iterative rules for updating pixels in a process of local competition and cooperation, which simultaneously exploits the data and the prior model for the image. They are used to solve global optimisation problems which meet the following criteria:

- an analytic solution is not available
- the cost domain has many local minima
- a grand tour is too costly.

The Bayesian approach provides a unified framework for image models, so avoiding the need to adopt *ad hoc* methods for each new problem.

1.2 Full pixel edge detection

1.2.1 Motivation for studying full pixel edge detection

One important application in image analysis is the location of edges between regions of different colour, known as *edge detection.* Edges convey a surprising amount of information about a scene, as illustrated by the human ability to interpret cartoons. The simplest case is a two colour problem, which arises in images that are essentially binary in nature. This occurs naturally when locating the interior and exterior of an object. Sometimes, it is sufficient to detect the edges between regions of different colours to the nearest pixel. In such cases, the edges are implicitly assumed to lie on the boundary between adjacent pixels.

1.2.2 Current methods for full pixel edge detection

Filters Filters enhance or emphasize certain features of images, by applying transformations based on sets of neighbouring pixels. The objective is to remove noise or enhance edges. Their main advantage is that they are fast and for simplicity, most filters are designed to be spatially invariant. There are two general classifications: linear and non-linear filters. A filter is linear if the values it outputs are a linear combination of its input values, otherwise it is a nonlinear filter.

Linear filters The output from *linear filters* is a linear combination of the values in the record. Linear filters are easier to analyse mathematically so, not surprisingly, the theory is much more developed. This class of filters can be studied in the spatial or frequency domain. There are many linear filters but we only mention the more common ones. We start with the box filter, which can be used efficiently to approximate the Gaussian filter. Both of these filters smooth the data, reducing noise. This helps edge detection filters, such as the Laplacian, because they are often sensitive to noise. It is possible to combine the filters to achieve the best of both worlds, such as the Laplacian-of-Gaussian filter.

Box filter This is simply a moving weighted average (MWA). So the output from a linear filter g_{ij} of size $(2p+1) \times (2p+1)$, with weights W_{kl} for $k, l = -p, \ldots, p$, is

$$g_{ij} = \sum_{\substack{k=-p \\ l=-p}}^{p} W_{kl} Y_{i+k,j+l}, \quad \text{for } i, j = p+1, \ldots, n-p,$$

where Y_{ij} denotes the record value in the i^{th} row and the j^{th} column. The output g_{ij} represents the reconstruction of the record Y, as a consequence of applying the filter.

For example, every pixel could be replaced by an average of the 3×3 window around it. Odd sized windows are typical and adjustments are needed at the boundary. This filter reduces noise but increases blurring. It is used to smooth images rather than detect edges.

Gaussian filter This filter has weights $\mathbf{W}$ derived from a bivariate Gaussian distribution, with variance σ^2,

$$W_{kl} = (2\pi\sigma^2)^{-1} \exp\left\{\frac{-(k^2+l^2)}{2\sigma^2}\right\}, \quad \text{for } k,l = -[3\sigma],\ldots,[3\sigma]. \tag{1.1}$$

Here, $[3\sigma]$ is the integer part of 3σ and it controls the amount of smoothing. The distribution is truncated for convenience. The Gaussian filter has a variety of appealing mathematically properties (Marr and Hildreth 1980). This filter is used to model blurring in Chapter 4. It can be approximated efficiently by repeatedly applying a MWA, which is a form of kernel regression (Scott 1992).

Laplacian filter Subtracting corresponding pixels in the box filtered image from the original image produces a new filter, called the *Laplacian filter.* It is a linear, second order differential operator, which produces a zero crossing at an edge. The Laplacian is

$$\begin{aligned} g_{ij} = \frac{\partial^2 Y_{ij}}{\partial x^2} + \frac{\partial^2 Y_{ij}}{\partial y^2} &\approx \sum_{\substack{k=-1\\l=-1}}^{1} W_{kl} Y_{ij} \\ \text{where} \qquad \mathbf{W} &= 1/3 \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \end{aligned} \tag{1.2}$$

Once again, the output g_{ij} represents the reconstruction of the record Y, as a consequence of applying the filter. One important property is that this filter can detect edges in any direction. In other words, the Laplacian is *isotropic* or rotationally invariant, so it is widely used in edge detection.

Laplacian-of-Gaussian filter Another common filter is formed by applying the Gaussian to reduce noise followed by the Laplacian to emphasize edges. This convolution of Equations (1.1) and (1.2) is called the *Laplacian-of-Gaussian filter* (LOG) (Marr and Hildreth 1980).

Oakley and Shann (1991) extend this method of edge detection, using Gaussian filtering calculated at optimally chosen points of interest. The accuracy is based on the filter size.

Non-linear filters All filters that are not linear are classified as *non-linear filters.* Non-linear filters are more diverse and difficult to categorise. They often do better than linear filters because they can reduce noise without blurring edges but they tend to be more complex and have less theoretical justification. Simple statistical examples include the moving median and the trimmed mean. To detect edges, some of the weights in the filter must be negative and, if the weights sum to one, the resulting image emphasizes the edges in the record.

Roberts filter A common method for finding edges is to apply a discrete operator which takes a maximum on an edge. One simple low pass, gradient operator which reaches a maximum at the edge is the *Roberts filter* (Roberts 1965). It approximates the magnitude of the gradient at the point (i, j) by using the sum of absolute differences of diagonally opposite record values,

$$g_{ij} = |Y_{ij} - Y_{i+1,j+1}| + |Y_{i+1,j} - Y_{i,j+1}|.$$

Prewitt and Sobel filters The maximum gradient at a point is

$$\sqrt{\left(\frac{\partial Y_{ij}}{\partial x}\right)^2 + \left(\frac{\partial Y_{ij}}{\partial y}\right)^2}.$$

These partial derivatives can be approximated in various ways. For example, the approximation

$$\widehat{\frac{\partial Y_{ij}}{\partial x}} = \frac{1}{6}(Y_{i-1,j+1} + Y_{i,j+1} + Y_{i+1,j+1} - Y_{i-1,j-1} - Y_{i,j-1} - Y_{i+1,j-1})$$

and

$$\widehat{\frac{\partial Y_{ij}}{\partial y}} = \frac{1}{6}(Y_{i+1,j-1} + Y_{i+1,j} + Y_{i+1,j+1} - Y_{i-1,j-1} - Y_{i-1,j} - Y_{i-1,j+1})$$

is known as *Prewitt's filter.* *Sobel's filter* is similar except that more weight is given to pixels near (i, j).

Because edge detectors require assumptions about the shape of the features being measured, such as straight lines verses circular edges, some authors use an array of edge detectors each sensitive to a different group of edge types. Glasbey and Horgan (1994, Chapter 3) offers an excellent introduction to filters.

Transforms If the low dimensional parametric description of the object is known, then a model fitting procedure can be used to find the parameter values for that object that best fit the data. The method works well even in the presence of noise.

For example, Hitchcock and Glasbey (1994) parametrise small objects, peanut kernels, using Fourier descriptors for the boundary and then optimise the parameters. They also describe an image of fibres using a network of splines.

Glasbey (1994) considers reversing the degradation caused by digitising images. Inference from binary images to binary data follows a two stage process: firstly from the lattice points to a binary image in continuous space and then inverting the blurring and removing noise. If objects, of a minimum size, have a specified shape and size except for one or two parameters then it is possible to estimate the unknown parameters and their standard error.

The Hough transform method replaces a search through image space with a search through Hough transform space (Atiquzzaman 1992). This requires a mathematical model of the object shape and the space to be searched may be large.

Contours An approach to perimeter estimation, due to Koplowitz and Bruckstein (1989), is based on contour reconstruction methods with subsequent measurement of the reconstructed boundary by counting the vertical and horizontal links lying on the pixel boundary around the object. Each link has length equal to the pixel size. The number of links would have to be adjusted to smooth this stepwise function. The adjustment is such as to yield an expected error of zero between the true and estimated length of straight boundaries.

Snakes Active contours (Kass, Witkin, and Terzopoulos 1987), also called snakes, is an approach where the boundary is represented by a fixed number of points. Unlike Friedland and Adam's star-shaped approach (see below), the points are not restricted to being of a fixed radius. Functions are maximised to include spatial smoothness and combine the boundaries of the image. To be more specific, a local gradient operator is first applied to the image. Then a smooth curve is found such that, the integral of the squared image gradient along the curve is maximised. Algorithms using this approach have concentrated on being fast and can usually produce answers in real time but this method will only find local optima, using a steepest decent procedure. Therefore, it needs a good starting point in order to avoid convergence to a local maximum.

In a similar spirit, O'Sullivan and Qian (1994) characterise object boundaries as areas where there is rapid change in the grey-level intensity of the image. They then use a function that measures the normalised image contrast between the inside and outside of a boundary to detect closed curves in the image.

Maximum entropy This is a Bayesian method that uses a non-spatial prior. It is especially useful if the background is mostly black with point sources of high intensity, such as arise in astronomy (Weir and Djorgovski 1991; Skilling and Bryan 1984). In these cases, the edges are the border between the black and white pixels. A substantial amount of computing is required.

Stochastic models A variety of models have been proposed that use a Markov random field to model images as part of a Bayesian approach to boundary detection at the *full* pixel level. The Bayesian approach to two dimensional, image reconstruction

was popularised by Grenander (1983), Geman and Geman (1984) and Besag (1986). Theoretical and practical applications have been pursued ever since. In general terms, this methodology treats the recorded image as numerical data, generated by a statistical model, involving both a stochastic component to allow for noise and a systematic component to describe the true scene under view. The corresponding likelihood is combined with a prior model for the true scene to draw inferences about the scene, based on the observed data, using Bayes' theorem.

These statistical, edge detection models often assume simply connected boundaries, sometimes called *dynamic contours*. They are found by iteratively making local changes to an existing boundary until convergence.

MRF edge elements Geman and Geman's paper suggested using edges between pixels to define a boundary around objects in an image. Such edges would lie either horizontally or vertically. Silverman, Jennison, Stander, and Brown (1990) and Silverman and Jennison (1991) re-examine the Geman and Geman model and propose edge penalties that are roughly independent of the pixel grid, based on geometrical features of the image. Then, if an edge exists between two contiguous pixels, they are not considered to be neighbours in the reconstruction.

Star-shaped boundary The star-shaped approach (Friedland and Adam 1989) represents boundaries as the distance from a centre point along a series of radii. The object being studied must be star-shaped around a centre point. A Bayesian approach combined with the simulated annealing algorithm can be used to find the optimal boundary by minimising an energy function. The energy function can also include spatial and time-dynamic aspects as well as data models. This methodology is explained in greater detail in Chapter 2. The resolution varies along the boundary because the distance between the radii increase with increasing distance from the centre point.

Stochastic boundary As an alternative to the snakes algorithm of Kass, Witkin, and Terzopoulos (1987) (see above), Storvik (1994) assumes that the boundary consists of a *stochastic* number of points with equal distance between adjacent points. The points are usually set to be one pixel apart so that the boundary consists of horizontal and vertical, linked edges. A global simulation scheme is used to find the optimal boundary. The model is easy to update, once an initial boundary has been found.

Helterbrand, Cressie, and Davidson (1994) offers an MCMC approach to identifying one-pixel wide closed boundaries, that is in the same spirit as Storvik (1994). The optimisation algorithm is restricted to searching over a class of closed boundaries so the final reconstruction is guaranteed to be closed. Because of this constraint, various configurations of edges are specifically excluded.

Constrained optimisation Geman, Geman, Graffigne, and Dong (1990) define edge sites and neighbourhoods to exist at a coarser scale than that of the record. This helps to prevent the profusion of 'micro edges' that may be suggested by apparently fuzzy boundaries. Certain configurations are banned but only gradually. This is achieved by steadily increasing the penalty for undesirable states. The model is also used for texture discrimination.

All these methods require some *a priori* information about the features being studied. Some methods define a mathematical formula for the shape of the boundary and then optimise the model parameters based on the observed data, such as the star-shaped algorithm and other algorithms that use Fourier descriptors. Other approaches do not require any assumption about the shape of the boundary but the cost is that a large optimisation problem must be solved. Each model or algorithm offers its own set of advantages and disadvantages. The most suitable choice depends on the purpose of the analysis. One common problem is that it is not immediately apparent how to extend these models to three dimensions.

1.3 Subpixel edge detection

1.3.1 Motivation for studying subpixel edge detection

The location and orientation of the pixel grid can have a noticeable effect on the shape of objects that are at the resolution limits of the recording sensor. This is because the reconstruction of a continuous planar image is piecewise constant on pixels, while boundaries in the image consist of horizontal and vertical edges lying between pixels. This approximation to the true boundary can result in a loss of information which may be quite noticeable for small objects, only a few pixels in size. Increasing the resolution of a sensor requires greater resources, which may be undesirable or not possible in some applications, due to technical or cost limitations. For example, an increase in intensity of illumination may damage a biological sample or in medical imaging the lowest possible exposure to X-rays is always desirable. So in such cases, minimising the amount of such exposure may be an important constraint. If not, perhaps the number of detectors could be increased by a factor of ten, say, in each of the x, y and z directions. However this would result in a thousand-fold increase in the cost and amount of subsequent processing.

We wish to reduce the distorting effect caused by the pixel grid. Resolving detailed features of an object is not possible if the object lies wholly within a single pixel but usually individual objects within an image extend across the boundaries between pixels. By expressing fairly mild assumptions about the underlying image in a mathematical model and combining this with the observed data, it is possible to reconstruct the image to a higher accuracy than the scanning resolution of the sensor. The motivation for such a model comes from the belief that the true, underlying image has a smooth, continuous boundary. To achieve this, we remove the restriction that edges should lie between pixels

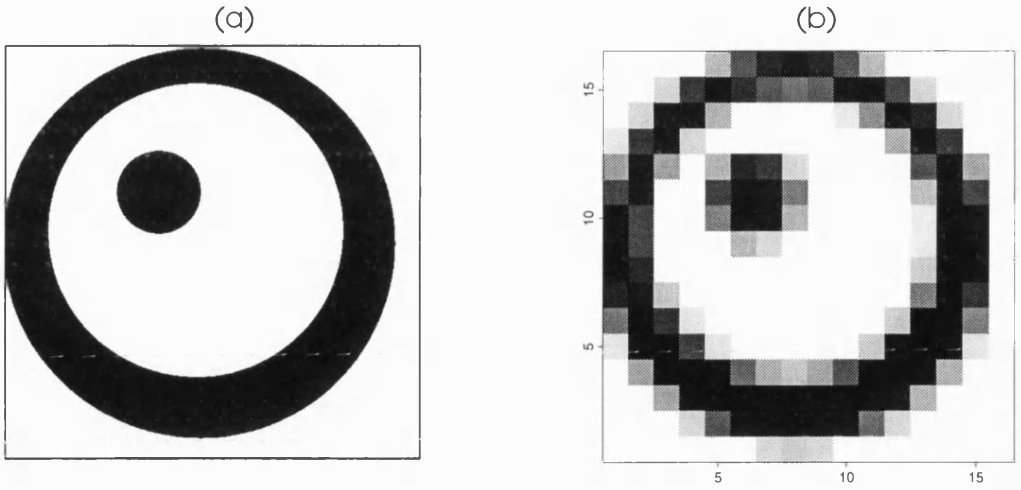

Figure 1-2: EFFECT OF PIXELATION ON A SMALL IMAGE. This figure shows the discretisation of a continuous binary image (Plot (a)) into a 16 × 16 grey-level pixel image and a reconstruction of the continuous image from the discretised record (Plot (b)). The effects of pixelation can lead to noticeable errors in edge length and surface area for objects that are small relative to pixel size.

and this leads naturally to a subpixel model. This process can be used to provide improved estimates of edge length and area of objects and to detect the presence and shape of objects that are only a few pixels in size. This procedure is called *subpixel resolution* and is the focus of this thesis.

Example 1.2. EFFECT OF PIXELATION ON A SMALL IMAGE.

For example, the continuous binary image shown in Figure 1-2 (a) appears as the discretised image, Figure 1-2 (b), when recorded on a 16 × 16 pixel grid. The level of grey in each pixel of the discretised image represents the proportion of black in that pixel in the original image. There is just a small amount of noise and no blurring in this example so almost all loss of information is due to the discrete nature of the sensor. Ideally, the properties or position of the pixel grid should have a minimum effect on the reconstruction of the true image.

We consider this toy image in greater detail in Chapter 4 (see page 64). □

1.3.2 Applications of subpixel edge detection

It is reasonable to ask why anyone would be interested in measuring data to a greater accuracy than the scanning resolution of the sensor. One way of answering this question is to consider the areas where subpixel models have found application. In fact, there are surprisingly many applications, such as:

- Measuring the size and shape of microscopic fibres (Hitchcock and Glasbey 1994). The authors study blurred, grey-level images of true binary scenes. Fourier descriptors are used to optimise the parameters for the boundary around small objects. A

network of splines is used to extend the technique to images of fibres. In Chapter 4, we apply a subpixel model to some microscopic data of a fungus that are similar to the data considered by Hitchcock and Glasbey (1994).

- Classification of subpixel vegetation cover (Jasinski 1990a; Jasinski 1990b; Jasinski and Eagleson 1990; Foody 1994).
- Remote sensing of active volcanos (Bhattacharya, Reddy, and Srivastav 1993).
- High precision measurement of component positions (Young 1987; Lyvers, Mitchell, Akey, and Reeves 1989). An edge operator based on two dimensional spatial moments is used with varying window size. A look-up table is used to correct for orientation and location or non-ideal edge profiles.
- Measuring the width and path of a laser beam (Szirányi 1992; Szirányi 1993). To recognise subpixel patterns in two dimensions or to measure laser beam diameters, a gray-level histogram of the objects examined is compared with the simulated histograms of different (in type or size) possible objects, and the shape or size is recognised on the basis of comparison.
- Locating and measuring blood vessel boundaries and diameters in cine coronary angiography (Sandor and Spears 1985). A Monte Carlo technique is used to estimate the effect of radiographic noise on the location of blood vessel boundaries in an image. The objective is to measure changes in blood vessel diameters in order to estimate the fraction of the vessel that is open for blood flow. The authors fit polynomial curves to a one dimensional cross section through the blood vessel using a Monte Carlo approach. This requires an assumption about the order of the polynomial to choose.
- Subpixel alignment in lithography (Gatherer and Meng 1992; Gatherer and Meng 1991).
- Pictorial data in the form of line drawings (Sriraman, Koplowitz, and Mohan 1989; Koplowitz and Sundar Raj 1987). A nonlinear, filtering algorithm is used to reconstruct piecewise linear curves such as polygonal figures.
- Subpixel deconvolution in astronomy (Weir and Djorgovski 1991) based on maximum entropy and a Bayesian model.

These applications rely on a variety of techniques that are often specific to the particular application in hand.

1.3.3 Current methods for subpixel edge detection

There are a wide variety of *ad hoc* methods for subpixel edge detection. For the most part, they are based on extensions to the full pixel methods outlined in §1.2. Some of the more common methods are:

Interpolation Interpolation is a simple method that produces fewer distortions and smoother results than simple pixel replication or sampling. The image is treated as a discrete approximation to a unknown, continuous object sampled at points on a lattice. To enlarge the reconstruction we could sample at a greater frequency but this is often neither feasible nor desirable. So we estimate the true image at inter pixel points by assuming smooth changes occur between the four nearest neighbour record values. This method is usually used in conjunction with other methods, such as filtering.

For example, Lorensen and Cline (1987) propose a 3D algorithm that effectively thresholds the data, which they call 'marching cubes'. It efficiently draws a surface through the 3D data by interpolating between neighbouring voxel values (see §5.1.2 on page 103).

Filters Because the sampled points are discrete, numerical approximations to the continuous filter are needed to get subpixel edge locations (Oakley and Shann 1991; Huertas and Medioni 1986). For example, filtering might first be applied. Then a second stage might be to interpolate between the discrete sampled points, by fitting a polynomial (Sandor and Spears 1985). Subpixel accuracy is then obtained by finding the zero crossing or maximum value, as appropriate, in the continuum.

Huertas and Medioni (1986) locate edges by finding zero-crossings in the convolution of the image with a Laplacian-of-Gaussian mask. The zero-crossings locate the edges and then it is assumed that in the neighbourhood around such points the image can be modelled by a polynomial. Subpixel values are then obtained by sampling the continuous function on a regular grid at the desired resolution.

Boult and Wolberg (1993) treated images as area samples, rather than the more conventional treatment of images as point samples. They formulate a process with the constraint that the approximate reconstruction is the exact reconstruction for some function in the allowable class of functions. The class of functions is based on the second derivative of the underlying continuous signal.

Filter methods provide fast subpixel algorithms because the filter is chosen to be simple. These methods do not require iteration or knowledge about the shape of the edges in the image. However, subpixel accuracy cannot be achieved without interpolation and the results may be distorted by noise. Amongst other things, this requires an assumption about the order of the polynomial to choose. In addition, an assumption about the edge filter width is required. Some allowance is also needed for the effect on accuracy, if edges are in close proximity but not connected.

Transforms Kiryati and Bruckstein (1991) show that if binary images of polygonal silhouettes are to be digitised then low-resolution gray-level sensors can potentially induce less ambiguity than high-resolution binary sensors.

For subpixel alignment in lithography Gatherer and Meng (1992) use a discrete Fourier transform.

Template matching An ideal edge is fitted to the observed record values, lying in some window, by matching some statistics calculated from the proposed edge and the record values or by maximising some joint statistic such as correlation. So the ideal edge is rotated, translated or rescaled to best fit the observed data where, for example, the ideal edge might be a step or ramp edge.

Because the shape of the object is assumed known, correlation can be used to determine the translation, rotation and scale parameters. Correlation is used because the cross-correlation function is smooth enabling interpolation to find the correlation peak (West and Clarke 1990), such as matching between ariel photographs. The ideal moments are calculated by integrating the ideal edge over a chosen window, subject to the sample moments being equal to the moments of the ideal edge.

This approach offers a closed form solution and requires no interpolation or iteration. However, it requires assumptions about the shape of the edge (e.g. the edge forms a straight line or is circular over the window) and the size of the window over which the edge is estimated. Also the method may be distorted by noise. Tabatabai and Mitchell (1984) discuss a method of moments algorithm and Lyvers, Mitchell, Akey, and Reeves (1989) discuss an extension to this algorithm which includes spatial information in the moment equations. Jasinski and Eagleson (1990) consider an application of the method in the classification of subpixel vegetation cover.

Stochastic models Jennison and Jubb (1988) and Jubb and Jennison (1991) introduce a stochastic model for the edge process. They first find the boundary around an object and then optimise the position of a sequence of linked line segments around the boundary to provide a subpixel estimate of the edge. The model favours patterns with smaller total edge length.

West and Clarke (1990) offer a review of several subpixel methods. These methods have much to offer, including considerable scope for further extensions and computational improvements. However, it is essential to be aware of their limits, as no single method is universally satisfactory. The methods that we outline in subsequent chapters are not replacements for the methods listed above but a further alternative to be added to an analyst's toolbox. Our models have their own limitations which must be recognised but they offer mathematically and computationally attractive features not found in the methods outlined above.

1.4 Broad view of this thesis

1.4.1 Objectives

The theory behind Bayesian image analysis has reached a certain maturity, so now much of the work in this area is focussing on applications. Such is the case in this thesis. We summarise our objectives as follows:

Reconstructing edges Our primary objective is to improve the quality of the restoration at a boundary, between two regions of different colour, by allowing each pixel to contain two colours.

The ways in which each pixel is allowed to be split leads to various models: in Chapter 3, each pixel is gradually refined by dividing it into discrete subpixels; the continuous analogue of Chapter 3 is to allow each pixel to be split in to two regions separated by a straight line and this is explored in Chapter 4. Due to recent improvements in technology, it now possible to record data in three dimensions. For example, the data might be recorded as a sequence of parallel and equally-spaced, two dimensional slices through a three dimensional object. In Chapter 5, we investigate how the model in Chapter 4 might be extended to incorporate three dimensional images.

We remove imperfections in the data, by designing a statistical model of the degradation and then search over a predefined class of reconstructions to find the reconstruction that strikes the best balance between the data and our prior beliefs about the true image.

Algorithmic design We structure all algorithms to be *strictly local* in nature. This makes the computational aspects of the algorithms easier to understand and visualise, and consequently to implement. It also means that the algorithms are potentially very efficient in their use of CPU time and storage requirements.

Adaptable models It is surprisingly difficult to describe mathematically the information in an image. Any models should be flexible enough to be applied to real data under various conditions. The sensitivity of the model to the assumptions made about the data should be considered and the model should be robust with respect to extreme behaviour.

Visualisation Compact representation of the objects in images is very desirable, most especially in three dimensions. We consider reconstructions that extract structure from the record in a concise form. This reduces the burden on the computer when viewing and storing the reconstruction. It also makes the model and algorithm more transparent and thus easier to understand.

A natural advantage of subpixel imaging is the ability to zoom in on features of interest. This requires the reconstruction to preserve detail. While there is a limit

to the level of detail available from any reconstruction, the techniques discussed are designed to detect small features in the data. The facility to rotate and zoom-in on the data is especially appealing with computer reconstructions of three dimensional objects, where motion is often essential to create the illusion of depth.

Measurement This is usually the final stage in the analysis. A compact representation of the structure in the observed data makes it easier to extract summary information. If the reconstructions preserve detail then more accurate estimates are possible. The basic building block is an estimate of the edge length in each pixel or the area of a surface through a voxel in three dimensions. From this, we can build up estimates of the area or volume of an object in an image and other statistics.

Overall, this thesis proposes a new initiative for statistical applications in the field of image analysis as well as some new computational developments. The techniques proposed could be seen as a contribution to the field of image filtering.

1.4.2 Outline

- Chapter 2 is concerned with the necessary background details that form the basic building blocks of subsequent chapters.
- Chapter 3 introduces a computationally feasible, discrete, two dimensional model for subpixel image reconstruction. Computational problems are discussed.
- Chapter 4 is concerned with a continuous, two dimensional model for subpixel image reconstruction. The fungal data set, shown in Figure 1-1 (a), is analysed to a subpixel level.
- Having established a continuous model and algorithm for two dimensions, Chapter 5 extends this algorithm to three dimensions and discusses the additional complications that arise. It also summarises the existing approaches to three dimensional image reconstruction.
- Finally, we draw some overall conclusions.

Each chapter finishes with a summary. The appendices contain material that is useful but not essential to the thesis.

1.4.3 Topics not covered

Naturally, this thesis does not cover every topic related to image reconstruction. It is informative to state explicitly which topics within this field are not discussed. Excluded topics include:

Dynamic images Most images are of static objects captured under laboratory conditions. It is assumed that there is time to prepare, process and analyse the data.

Real-time, automatic processing of images involves studying motion, three dimensional modelling and real-time processing. These are topics from the field of computer vision that are beyond the scope of this thesis.

Images that vary smoothly Objects which vary smoothly across the image are not considered in this thesis. All the examples analysed in later chapters show abrupt changes in pixel values at the edge of the object, which we interpret as object boundaries.

Indirectly observed images We exclude indirectly observed images, such as Positron Emission Tomography (Rosenfeld and Kak 1982). However, we do need to consider inverting a blurring matrix, if the image is blurred.

Mathematical morphology This field of study uses set theory to apply a series of operations to an image to define the shape of the objects that it contains. Originally, it was applied to binary images only. More recently, morphology has been extended to deal with grey-level images. Serra (1982) is the seminal reference for this subject.

Object recognition This thesis is restricted to low-level analysis such as the removal of noise and blurring. The main problem with pixel-level image models is that inference based on them can only refer to statements about the pixels of the true scene. However because MCMC is used to implement the image models, posterior probabilities of arbitrary events defined by the pixel values, such as boundary length, can be estimated. This is discussed in Chapter 2. We do not attempt high-level analysis from the field of artificial intelligence, such as object recognition, as has been done using template matching (Green 1995a, §3.3).

Registration Subpixel registration is a related and important topic in this field. It is concerned with accurately aligning two or more separate images of an object to allow consistent comparison to be made between two or more images (Tian and Huhns 1986; Mort and Srinath 1988).

It is assumed that the data have already been registered and, for the 3D data considered in Chapter 5, this is guaranteed because of the recording method used by the sensor.

Segmentation Segmentation means dividing the pixels in an image into categories, which correspond to different objects or parts of an object. The objective is to move from individual pixels to considering sets of pixels as objects in their own right.

The simplest segmentation algorithm is to threshold the record, by classifying each pixel as lying above or below the threshold. The threshold might be the median record value in the image.

Some segmentation algorithms are often quite complex. For example, the Hough transform is a subpixel segmentation algorithm and the snakes algorithm operates

at the full pixel level (see §1.2.2 on page 8). Kent and Mardia (1988) introduce fuzzy membership models in order to classify pixels in LandSat data.

Although we are not concerned with segmentation itself, it is in fact complementary to edge detection since edges can be used to break up images into different regions.

1.5 Summary of chapter

The purpose of this chapter is to provide motivation to explore the subject of subpixel image analysis in greater detail. To achieve this:

- Image analysis is defined and some example images are shown. The field of image analysis is broken down into categories.
- Potential areas of application are listed.
- Full pixel edge detection is explained and commonly used methods are summarised.
- Reasons for subpixel edge detection and possible applications are discussed.
- Some existing subpixel methods are outlined, including:
  - interpolation
  - filters
  - transforms
  - template matching
  - stochastic models.
- Objectives for this thesis are stated:
  - accurate reconstruction of edges in an image
  - efficient algorithmic design
  - flexible statistical models
  - fast visualisation of the reconstruction.
- The thesis is outlined.
- Topics not covered in this thesis are acknowledged.

Chapter 2

Bayesian models and inference

The sciences do not try to explain, they hardly even try to interpret, they mainly make models. A model means a mathematical construct which, with the addition of certain verbal interpretations, describes observed phenomena. The justification of such a mathematical construct is solely and precisely that it is expected to work.

John von Neumann

Models are to be used but not to be believed.

Henri Theil

2.1 Basic assumptions about image data

In image analysis, there is often prior knowledge about the features of objects in the image. Other assumptions can be derived from the data directly, such as estimating the colours, noise and blurring in the likelihood. In this thesis, we make assumptions about the form of the true image through the prior model. To reconstruct the true image to subpixel accuracy, we make the following assumptions:

1. The image can be modelled by a Gibbs distribution.
2. Objects in an image are defined as regions displaying considerable homogeneity, and have smooth boundaries.
3. The regions are large, so that locally the image has at most two colours, perhaps with a few exceptions.

For the first of these assumptions, a stochastic Bayesian approach to image analysis is used to incorporate prior assumptions about observed and unobserved image features, such as the detection, linking and smoothing of edges. These prior assumptions are incorporated into a mathematical model through a prior probability distribution on the set of possible images. Any difference between a given reconstruction and the observed data is reflected through a separate model for the degradation. Typically, the degradation arises from noise and blurring due to sensor imperfections. Consequently, there are many different images that could have produced the observed data. To help to avoid this instability, assumption two states that object boundaries are assumed to be smooth, in some sense. However,

we do not require a parametric formula for the whole boundary. Instead, we implement this assumption by imposing a smoothness constraint, which says that pixels which are in close proximity are likely to be similar in colour. The third assumption ensures that within each region the image is essentially binary in nature. If the edge length or area of an object are the statistics of interest then it is sufficient to consider the object of interest to be lying on a contrasting background, so that there are only two colours or levels in the image. This assumption eases the computational burden and is often encountered in subpixel resolution modelling.

Remark.

- Note that we make no major assumptions about the number, shape or size of objects in the true image, except that the objects are at least one or two pixels in size, do not overlap and have sharp edges.
- The models make explicit allowance for noise and blurring, in the observed record.
- For convenience, we colour the foreground as black and the background colour as white. They are denoted by b and w, respectively.

  In principle, we would like to allow any division of the image between colours b and w in continuous space. However, our methods rely on the assumption that regions of a particular colour are at least a few pixels in size and that the boundaries between areas of opposite colour are smooth. But we observe that our methods have the potential to be extended to images containing several colours. The basic algorithm can be applied within any region containing just two colours and additional features are needed only in small regions where three or more colours meet. This is likely to occur at only a few pixels in an image. An alternative proposal is outlined in Chapter 6.

2.2 The Bayesian model

2.2.1 Some notation

Prior The true image is denoted by $X \stackrel{\text{def}}{=} (X_1, X_2, \ldots, X_n)$ and it is treated as a random variable. Let the density for the true image X be $f_X(x)$, where $x \in \Omega$ and Ω is defined to be the state space of true scenes. The density $f_X(x)$ is the *prior* distribution. The random variable X_i is the colouring of pixel i. The marginal density of X_i is written $f_{X_i}(x_i)$. In full pixel models, X_i is usually a univariate distribution but in our subpixel models it is more complex. We refer to an estimate of the image as a *reconstruction.*

While the underlying, true image is usually continuous, we normally assume that it consists of pixels laid out on a regular lattice. This approximation is usually adequate for full pixel analysis but for subpixel reconstructions greater care is needed to define the state space for each model.

Likelihood Let the probability density of the observed image or *record* Y given $X = x$ be $f_{Y|X}(y,x)$ and let $f_Y(y)$ denote the marginal density of Y. The density $f_{Y|X}(y,x)$ is the *likelihood* distribution. Although Y is a random variable, it is fixed by observation, once the data has been observed.

Posterior Let the probability density of the true image given the record be $f_{X|Y}(x,y)$. The density $f_{X|Y}(x,y)$ is the *posterior* distribution.

2.2.2 Bayesian inference

The methodology used in this paper is based on the Bayesian approach to two dimensional image reconstruction proposed by Grenander (1983), Geman and Geman (1984) and Besag (1986). In a Bayesian analysis, inference about X is based on the posterior density of X given $Y = y$,

$$f_{X|Y}(x,y) \quad = \quad f_{Y|X}(y,x)f_X(x)/f_Y(y) \quad \propto \quad f_{Y|X}(y,x)f_X(x), \qquad x \in \Omega. \tag{2.1}$$

Due to the high dimensionality of x, standard analytical, numerical or simulation methods are not adequate. Typically in spatial statistics $f_X(x)$ is known up to proportionality only. We are mainly interested in the posterior distribution $f_{X|Y}(x,y)$. For example, we might compute properties of $f_{X|Y}(x,y)$, such as the expectation $E_{f_{X|Y}}(k(X))$ of some function k under $f_{X|Y}$,

$$E_{f_{X|Y}}(k(X)) = \sum_{x \in \Omega} k(x) f_{X|Y}(x,y).$$

If k is an indicator function, probabilities of specified events are possible. Interesting properties of a reconstruction are not confined to just expectations. In §2.4, we consider which estimator has the most appropriate properties for subpixel imaging.

2.2.3 The prior distribution

We want the prior model for X to produce realisations which contain smooth boundaries so that the reconstruction itself will tend to have smooth boundaries. This can be achieved if pixels which lie close to each other on the lattice tend to have similar colourings. It is also reasonable to assume that pixels which are spatially separated on the lattice are independent, given the other pixels in the image. A *Markov random field* probability model (MRF) is a model that is often used to reflect both of these features. So it is used in image analysis to introduce prior knowledge about the image. Its key feature is the local dependence of an individual pixel on the pixels in a neighbourhood around it. Before defining a MRF, we introduce some set notation to denote specific sets of pixels within an image.

Definition 2.1. We say that two pixels are neighbours if they are near each other, in some sense. Denote a *neighbourhood* relation on $x_i, x_j \in X$ by $i \sim j$.

The relation $\sim$ (tilde) is symmetric, so $i \sim j \Rightarrow j \sim i$. A standard condition is that x_i is not a neighbour of itself, $i \not\sim i$. The implication of this definition is that the neighbourhood structure is quite local but only if the neighbour relation is appropriately defined. Denote the neighbours of pixel i by δ_i.

Definition 2.2. The set of four pixels horizontally and vertically adjacent to pixel i is defined to be the *first order neighbourhood* δ_i^1 of pixel i.

Definition 2.3. The set of four pixels diagonally adjacent to pixel i is defined to be the *second order neighbourhood* δ_i^2 of pixel i.

We use second order neighbourhoods only in Chapter 3.

Definition 2.4. A *clique* is either a set of pixels, all of whom are neighbours, or a singleton pixel. Define C to be the set of all these cliques and define X_{-i} to be the set of variables X excluding pixel i.

Now, we are ready to discuss Markov random fields.

Model 2.1. MARKOV RANDOM FIELD (MRF). *A MRF is defined by two conditions. Firstly, every permitted reconstruction is considered,*

$$\Pr(X = x) > 0, \quad \forall\, x \in \Omega. \tag{2.2a}$$

Secondly, the colouring for pixel i is conditionally independent of all other pixels, given the colouring of its neighbours δ_i,

$$\Pr(X_i = x_i \mid X_{-i} = x_{-i}) = \Pr(X_i = x_i \mid X_j = x_j, j \in \delta_i). \tag{2.2b}$$

A theorem due to Hammersley and Clifford provides a consistent, general form for the joint p.d.f. of a MRF, given the conditional probabilities in Equation (2.2).

Theorem. HAMMERSLEY-CLIFFORD. *The general form of the probability density function (p.d.f.) that satisfies the two conditions for a MRF, in Equation (2.2), is*

$$f_X(x) = \Pr(X = x) = k \exp\{-U_X(x)\}, \quad \text{where } x \in \Omega,\ U_X(x) = \sum_{c \in C} U_c(x) \tag{2.3}$$

and $U_c(x)$ depends only on the pixels in the clique c. The constant k ensures that f_X integrates to one but its value need not be known to implement our methods.

Proof. See Besag (1974) and Besag (1986). □

The term $U_X(x)$ is sometimes called the *energy function* (or *cost function* or *penalty*) of the image X. Equation (2.3) is sometimes called a *Gibbs distribution.*

Random field models are derived historically from models in statistical physics. Developments in the image model and the design of algorithms that use this model have proceeded in parallel. A MRF can be summarised succinctly as a non-directional, non-causal, conditionally independent structure. In practice, it is built up by:

1. Specifying a neighbourhood system.
2. Deciding the cliques associated with that neighbourhood system.
3. Determining the energy function associated with each clique. The MRF is *homogeneous* if the energy function does not depend on the position of the clique.

The outstanding problem is how to perform any computation with a MRF model. The answer is to use Markov chain Monte Carlo methods but the details are deferred until §2.3.

The prior distribution for this thesis

The prior for this thesis is based on a MRF and so it takes the form shown in Equation (2.3). The choice of neighbourhood system and the specification for $U_c(x)$ decides the behaviour of the prior for X. The specific details about the prior are different in each chapter, as the models change, so we defer the details until then. For now we use the general form of the prior to specify the posterior, in Equation (2.6).

2.2.4 The likelihood function

The observed data $Y = (y_1, y_2, \ldots, y_n)$ are recorded on a regular lattice of points $\{z_i \in \Re^2 : i = 1, 2, \ldots, n\}$. The sensor's output at each Y_i represents the average intensity within that pixel and is assumed to be proportional to the area covered by the object in that pixel. Due to imperfections in the recording sensor, the record Y may be degraded by blurring or additive noise.

Model 2.2. LIKELIHOOD. *The likelihood model is*

$$y_i = h_i(x) + e_i, \quad \textit{for } i = 1, \ldots, n, \tag{2.4}$$

where e_i is independent, additive sensor noise and $h(x)$ is the vector of mean values $E(Y)$ when $X = x$, taking into account blurring, if necessary. □

Example 2.1. CONTINUOUS 2D DEGRADATION MODEL.
In 2D, one possible additive Gaussian model for the degradation model is

$$Y_i \stackrel{\text{def}}{=} \iint x(z) b(z_i, z) dz + e_i,$$

where $x(z)$ is the value of the true image at the point $z \in \Re^2$, $b(z_i, z)$ is a blurring function which decreases in value as the distance between z and the point z_i increases and e_i is

independent, additive, Gaussian sensor noise, with variance σ^2. This particular model is considered in §4.2.2 on page 70. □

In general, the likelihood distribution of the signal y given the true image x for the Gaussian model is

$$f_{Y|X}(y,x) \quad \propto \quad \exp\{-(2\sigma^2)^{-1}\|y-h(x)\|^2\}. \tag{2.5}$$

The specific formulae for y and $h(x)$ depend on the model used, so we defer the details to later chapters. A simple example is given in §2.6.

Blurred data

The function $h(x)$ may include blurring. When blurring is present, pixels around the border of the image are only observed indirectly through their contributions to the records of neighbouring pixels. In this case, there are fewer elements in the record Y than there are pixels in the image X. We assume that the blurring is shift invariant. This spatial stationarity, usually assumed in image models, removes the need for experimentation with proposal distributions tailored to individual variables, representing individual pixels (see §2.3.7 on page 32). We assume the blurring coefficients $b(z_i, z)$ and the noise variance σ^2 are known or can be estimated from the data.

2.2.5 The posterior distribution

The posterior distribution is formed by substituting Equations (2.3) and (2.5) into Equation (2.1) to get

$$f_{X|Y}(x,y) \quad \propto \quad \exp\{-U_X(x) - (2\sigma^2)^{-1}\|y-h(x)\|^2\}, \qquad x \in \Omega. \tag{2.6}$$

The posterior distribution is still a Markov random field but with larger neighbourhoods than in the prior because of blurring. Although this implies that the posterior distribution depends on Y, this is fixed by observation. This posterior distribution is the basis of statistical inference given the record Y.

Definition 2.5. When we refer to the *energy of an image*, we mean the energy associated with the posterior distribution $f_{X|Y}(x,y)$ which is the quantity

$$U_{X|Y}(x) = U_X(x) + (2\sigma^2)^{-1}\|y-h(x)\|^2. \tag{2.7}$$

We consider methods for estimating the true image X, which require sampling values of X from $f_{X|Y}(x,y)$ or finding the value that maximises $f_{X|Y}(x,y)$. These are substantial computational problems, which are tackled using Markov chain Monte Carlo algorithms.

2.3 Markov chain Monte Carlo algorithms

2.3.1 Definitions

To discuss Markov chain Monte Carlo algorithms (MCMC), we need the following definitions:

Definition 2.6. A stochastic process $\{X^{(t)} : t = 1, 2, \dots\}$ is said to have the *Markov property* if future events depend on the current status and not the past. The discrete case is expressed as

$$\Pr\{X^{(t+1)} = x^{(t+1)} \mid X^{(0)} = x^{(0)}, \dots, X^{(t)} = x^{(t)}\} = \Pr\{X^{(t+1)} = x^{(t+1)} \mid X^{(t)} = x^{(t)}\}. \tag{2.8}$$

Definition 2.7. A *Markov chain* M is a discrete-time, time-homogeneous process with a countable state space which satisfies Equation (2.8).

Definition 2.8. The Markov process can be defined by one-step *transition probabilities*

$$q(x, x') = \Pr(X^{(t+1)} = x' \mid X^{(t)} = x).$$

These transition probabilities can be collected into a matrix called the *transition matrix*, $Q = \{q(x, x') : x, x' \in \Omega\}$.

Definition 2.9. A Markov chain is defined as *irreducible* if we can reach every state from any given state.

Definition 2.10. An irreducible Markov chain is *aperiodic* if it has a state i of period 1, where the *period* D_i of state i is defined to be

$$D_i = \text{g.c.d.}\left\{t : \Pr(\text{getting from } i \text{ to } i \text{ in } t \text{ steps}) > 0, \text{for } t \in \mathcal{N}\right\}.$$

State i is *periodic* if $D_i > 1$.

Definition 2.11. The transition matrix Q of a Markov chain is said to satisfy *detailed balance with respect to a distribution* π_X if

$$\pi_X(x)Q(x \to x') = \pi_X(x')Q(x' \to x) \quad \forall\, x, x' \in \Omega. \tag{2.9a}$$

The *general balance* condition is satisfied iff

$$\pi_X(x) = \sum_{x' \in \Omega} \pi_X(x')Q(x' \to x), \qquad \forall \quad x \in \Omega. \tag{2.9b}$$

The detailed balance condition is also called *local balance* or *time reversibility.* Detailed balance implies general balance and it is often easier to work with.

Next we state, without proof, that the Markov chain will converge to a unique limiting distribution, under suitable conditions. A limiting distribution of a Markov chain is a distribution over the allowable states of the chain which is maintained as the chain undergoes transitions.

Theorem. ERGODIC. *If $X^{(t)}$ is an aperiodic, irreducible and finite Markov Chain, there exists a unique limiting distribution π_X satisfying*

$$\pi_X(x) = \sum_{x' \in \Omega} \pi_X(x') Q(x' \to x), \qquad \forall \quad x \in \Omega.$$

and $\Pr(X^{(t)} = i) \to \pi_X(i)$, *where* $i = 1, \ldots, n$, $\pi_X(i) > 0$, $\sum_{i=1}^{n} \pi_X(i) = 1$ *and* $t \to \infty$.

Proof. See Feller (1968). □

The limiting distribution π_X is also called the *ergodic, equilibrium, invariant* or *stationary distribution* for the transition matrix Q. It follows that to sample from the equilibrium distribution $\pi_X(x)$, we can run a Markov chain with transition matrix Q satisfying the equations in (2.9) until the chain appears to have settled down to equilibrium.

Although the above theory has been presented in terms of countable state spaces, it also holds for non-countable state spaces.

2.3.2 Constructing a MCMC algorithm

Suppose we wish to sample from a distribution $\pi_X(x)$. For example, in image analysis we want to sample from the posterior distribution $f_{X|Y}(x, y)$, given by Equation (2.6). Ideally we want a sequence of independent realisations from $\pi_X(x)$. In some cases such as the multivariate Gaussian, it may be possible to sample directly from $\pi_X(x)$ using a transformation of uniformly distributed variables. In other cases, rejection sampling may be used (Smith and Gelfand 1992). In more complicated cases, we resort to the ergodic theorem to generate a sequence of dependent realisations from a single, finite-state, discrete-time Markov chain, which we design so that its limiting distribution is $\pi_X(x)$.

We construct Markov chains in which a single element of X is updated at each step. In image analysis, this corresponds to a process in which one pixel at a time is updated and the distribution of the updated value is conditional on its neighbours. If the Markov chain is irreducible and aperiodic then the limiting distribution of this chain is the posterior distribution in Equation (2.6). So we can get a Markov chain with the posterior distribution as its limiting distribution. Simulating from this chain causes the chain to wander through the states x, visiting each with probability $f_{X|Y}(x, y)$.

Example 2.2. EXPECTATIONS UNDER MCMC.
Suppose we want to construct a chain M and then use its empirical average $\bar{k}_l$ to approximate $E_{\pi_X(x)}(k)$, where $\bar{k}_l$ is calculated from a long, partial realization $x^{(1)}, x^{(2)}, \ldots, x^{(l)}$ of the Markov chain. If $x^{(1)}$ has distribution $\pi_X(x)$ then the general balance condition

$\pi_X(x)\,Q = \pi_X(x)$ means that the marginal distribution of each subsequent $x^{(t)}$ is also π_X. This ideal condition is rarely attainable for the problems under consideration. Instead, we use an arbitrary starting point from which we collect a sample. For assessing simulation errors, the Markov chain is usually run for some time, called the *burn-in period*, before collecting statistics so that $x^{(t)}$ should then have distribution rather close to π_X. Geyer (1993) advocates using a single long run rather than several short ones. □

Having established that certain Markov chains converge to a unique limiting distribution, it is still necessary to construct the transitions such that the limiting distribution is the desired posterior distribution. There are several MCMC algorithms for updating just a subset of the variables in x, x_A say, in accordance with detailed balance. We consider the Hastings algorithm, the Metropolis algorithm and the Gibbs sampler, in the next section.

2.3.3 MCMC algorithms for sampling from a distribution

The Hastings algorithm

Hastings (1970) provides a general form of MCMC algorithm. The objective is to create a Markov chain with limiting distribution $\pi_X(x)$ for $x \in \Omega$.

Algorithm 2.1. HASTINGS.

- *Start with a proposal matrix P. P is any convenient matrix, which defines an irreducible and aperiodic Markov chain.*

- *Set the transition probabilities Q to be*

$$Q(x \to x') = \begin{cases} P(x \to x')\alpha(x \to x') & x \neq x', \\ 1 - \sum_{x'':x''\neq x} P(x \to x'')\alpha(x \to x') & x = x', \end{cases} \tag{2.10a}$$

$$\text{where} \quad \alpha(x, x') = \min\left\{\pi_X(x')P(x' \to x)/\pi_X(x)P(x \to x'), 1\right\}. \tag{2.10b}$$

Then Q has the limiting distribution π_X. □

Proof of convergence. To see that Q has limiting distribution π_X we need to confirm detailed balance,

$$\pi_X(x)Q(x \to x') = \pi_X(x')Q(x' \to x).$$

Case 1 $x = x'$**:**

Nothing to be done.

Case 2 $x \neq x'$ **and** $\pi_X(x')P(x' \to x) \geq \pi_X(x)P(x \to x')$**:**

$$\begin{aligned}\pi_X(x)Q(x \to x') &= \pi_X(x)P(x \to x')\alpha(x \to x') \\ &= \pi_X(x)P(x \to x'), \qquad \text{because } \alpha = 1. \\ \text{Also } \pi_X(x')Q(x' \to x) &= \pi_X(x')P(x' \to x)\alpha(x' \to x) \\ &= \pi_X(x')P(x' \to x)\big(\pi_X(x)P(x \to x')/\pi_X(x')P(x' \to x)\big) \\ &= \pi_X(x)P(x \to x').\end{aligned}$$

Case 3 $x \neq x'$ **and** $\pi_X(x')P(x' \to x) < \pi_X(x)P(x \to x')$**:**
Repeat the previous case but interchange x and x'. □

The irreducibility of Q follows from that of P. The matrix Q is aperiodic since $Q(x \to x) > 0$, for at least one state x in Ω. The matrix P can be anything so we choose something easy. However α depends on π_X, so we need to be able to calculate $\pi_X(x')/\pi_X(x)$ for any $x, x' \in \Omega$ with $P(x \to x') > 0$. We set up P so that the only transitions from x to x' are those for which $\pi_X(x')/\pi_X(x)$ are easy to calculate.

Definition 2.12. In the context of image analysis, we usually assume $Q(x \to x') = 0$ unless x and x' differ at most at only one pixel, $x_{-i} = x'_{-i}$ for some pixel i. This is called *single site updating.* Occasionally, we update several pixels simultaneously and this is called *multiple site updating.*

Definition 2.13. A *sweep* of the image consists of working systematically once through each of the n pixels in an image.

We select the sequence of pixels to be updated, from a raster scan of the image.

Definition 2.14. A *raster scan* visits the pixels in an image, going from left to right and from the top to the bottom of the image, starting at the top-left. (We define a raster scan of a 3D volume on page 126 of Chapter 5.)

We usually talk about visiting all the pixels in an image rather than looping over subscripts and for this we use the phrase 'sweeping the image'.

Another way to view the Hastings algorithm is to sweep systematically through a sequence of transitions, $Q^{(i)}$ for $i = 1, \ldots, n$, where each $Q^{(i)}$ is of the form given in Equation (2.10), for some $P^{(i)}$. The transition $Q^{(i)}$ updates element i only.

The Metropolis algorithm

The Metropolis algorithm is another single-site update procedure. It was proposed by Metropolis et al. (1953) for simulating the evolution of a solid to thermal equilibrium. Let P be symmetric,

$$P(x \to x') = P(x' \to x) \quad \forall\, x, x' \in \Omega$$

then $\alpha(x \rightarrow x') = \min\{1, \pi_X(x')/\pi_X(x)\}$. The transition matrix Q can be derived as follows:

Algorithm 2.2. METROPOLIS.

1. *Let $x^{(t)}$ be the state at time t. Set $t = 0$.*
2. *Generate a proposal state x' according to P, $\Pr\{X = x'\} = P(x^{(t)} \rightarrow x')$, for $x' \in \Omega$.*
3. *Accept x' with probability $\alpha(x^{(t)}, x')$ otherwise remain at $x^{(t)}$. So*

$$X^{(t+1)} = \begin{cases} x' & \text{with probability } \alpha(x^{(t)}, x'), \\ x^{(t)} & \text{with probability } 1 - \alpha(x^{(t)}, x'). \end{cases}$$

4. *Increment t. Goto Step 2.* □

In image analysis terms, we select a new proposal uniformly from the range of possible pixel values, when updating the i^{th} pixel. The new proposal is accepted with probability

$$\alpha(x, x') = \min\{1, \pi_X(x')/\pi_X(x)\}. \tag{2.11}$$

The equilibrium distribution is unique if $Q(x \rightarrow x') = \alpha(x \rightarrow x')P(x \rightarrow x')$ and a sufficient condition for convergence is being able to move from any state to any other (Ripley 1987).

The Gibbs sampler

The Gibbs sampler consists of sampling sequentially from the conditional distributions of each variable given all the others. The normalisation to a proper distribution is simple and MCMC simulation is easy using the Gibbs sampler, if the distribution is discrete and takes few values. It was popularised by Geman and Geman (1984) and is referred to as the 'heat bath' in statistical physics.

Algorithm 2.3. GIBBS SAMPLER.

1. *Let $t = 0$. Start with any point in the multivariate distribution $x^{(t)} = x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$.*
2. *Sweep through the n elements of $x^{(t)}$ to generate $x^{(t+1)}$.*
   - *Sample $x_1^{(t+1)}$ from $\Pr(x_1 \mid x_2^{(t)}, \ldots, x_n^{(t)}, Y)$*
   - *Sample $x_2^{(t+1)}$ from $\Pr(x_2 \mid x_1^{(t+1)}, x_3^{(t)}, \ldots, x_n^{(t)}, Y)$*
   - $\vdots$ $\qquad\qquad$ $\vdots$
   - *Sample $x_n^{(t+1)}$ from $\Pr(x_n \mid x_1^{(t+1)}, x_2^{(t+1)}, \ldots, x_{n-1}^{(t+1)}, Y)$*
3. *Increment t and goto Step 2.*

The vectors $x^{(0)}, x^{(1)}, \ldots, x^{(t)}, \ldots$ *are a realisation of a Markov chain. The transition probability* Q *from* x *to* x' *is*

$$Q(x \to x') = \Pr(x'_1 \mid x_2, \ldots, x_n, Y) \Pr(x'_2 \mid x'_1, x_3, \ldots, x_n, Y)$$
$$\Pr(x'_3 \mid x'_1, x'_2, x_4, \ldots, x_n, Y) \ldots \Pr(x'_n \mid x'_1, \ldots, x'_{n-1}, Y) \quad \square$$

This completes a sweep of the Gibbs sampler. It produces a whole new image after each sweep by updating the current value of the reconstruction at each pixel with one sampled from the conditional distribution for the colour of that pixel.

To put the algorithm in the context of image analysis, let $x, x' \in \Omega$. The transition probability $Q^{(i)}(x, x')$ of moving from x to x' by changing a single pixel x_i is proportional to the probability of x'_i under the conditional density of X_i given that $X_{-i} = x_{-i}$, $\pi_{X_i|X_{-i}}(x'_i, x_{-i})$. That is, if $x'_{-i} = x_{-i}$, the probability of moving from x to x' is

$$Q^{(i)}(x, x') = \begin{cases} \pi_X(x') / \sum_{x'': x''_{-i} = x_{-i}} \pi_X(x'') & x'_{-i} = x_{-i}, \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

and this move is always accepted. The algorithm works by applying these transition matrices in sequence, such as a raster scan. It can be shown that π_X is the stationary distribution for each $Q^{(i)}$ and that π_X is the stationary distribution for Q.

2.3.4 Comparisons between MCMC sampling algorithms

In the chapters that follow all three algorithms are used, so we now consider their relative merits.

The Gibbs sampler is a special case of the Metropolis algorithm which in turn is a special case of the Hastings algorithm. For all three algorithms, a neighbouring state is generated and a decision is made to accept the move, or not, from the current state to the newly, generated state. Generally, a *neighbouring state* is an image which differs from the current image by a single pixel. Chapter 3 also considers the possibility of changing more than one pixel at a time.

Choosing between the three MCMC methods depends on the application. Gibbs is useful if the marginal distribution takes few values (less than about 30) and the algorithm assumes that the marginal distributions are cheap to compute; otherwise this algorithm is computationally expensive. Hastings and Metropolis methods update pixels singly by choosing between the current and an alternative proposal. They work well with either discrete or continuous variables. Metropolis is easy to code but Hastings is a little more difficult because the transition matrix is no longer symmetric, so extra code is required. However, this allows greater freedom when designing algorithms that are more efficient and hence speed convergence. This extra feature is especially important with 3D images as

the greater amount of physical space translates into state spaces which grow exponentially with dimensionality. This point is discussed in detail in Chapter 5.

2.3.5 Characteristic features of MCMC algorithms for sampling the posterior image distribution

An analytic approach to sampling from the posterior is not feasible because of the high dimensionality of Ω. We resort to simulation to reflect and assess the uncertainty inherent in image models. We seek efficient and easy to code algorithms. The main advantage of simulation over analytical or asymptotic methods is its flexibility for non-standard cases and its ability for almost exact solutions for arbitrary functionals of the process. This is an important feature when dealing with image data. Simulation is used in estimating expectations, variances or modes, over some defined multivariate distribution. For example, the expectation might be probabilities or vectors of probabilities. The posterior distribution is often stated up to a normalising constant.

Conditional independence

The key to construction of highly structured stochastic systems is the concept of conditional independence, where each variable is related conditionally or locally to only a few other variables. We use a local conditional structure to economically model the spatial structure in the true scene. This allows the model to exhibit global complexity even though the model has a simple local structure. For example, conditional independence can be represented graphically by drawing random variables as nodes and the links between them indicate local conditional independence relations. This leads to modular components which are based on local computation.

Spatial structure

Secondly, a key feature of image data is their spatial structure. The conditional independence structure inherent in MCMC methods reflects this spatial structure while still producing tractable algorithms. This means that pixels in an image which are spatially near to each other are treated as neighbours in the conditional independence structure. So instead of producing a sequence of realisations from the multivariate distribution of interest, MCMC produces a single realisation of a Markov chain whose limiting distribution is that multivariate distribution. A key feature of this method is to work with conditional distributions so variables are updated one at a time. This slows the process down and leads to correlated results but each step in the algorithm is easier to compute.

Stationarity

Thirdly, to make image models even more tractable, they are often defined to be spatially stationary. This leads to efficient MCMC algorithms because the calculations at each pixel are very similar, resulting in compact code. This suggests that MCMC algorithms may

be implemented on parallel machines to produce real time answers (Grenander and Miller 1994). In §4.2.2 for instance, a likelihood function for the data relies on the blurring in the image being stationary, to simplify the calculations.

For these three reasons (Green 1994b), MCMC is a natural mechanism for sampling from both the prior and posterior distributions of image models. However, there is no substitute for a thorough qualitative understanding of the target distribution. This 'expert knowledge' is a prerequisite for reliable design of MCMC algorithms.

2.3.6 Advantages of MCMC sampling algorithms

Flexibility

MCMC is flexible. For example it can be used to estimate confidence intervals, to deal with non-standard priors and likelihoods that arise with missing data, to simulate systems that can only be observed indirectly, or to assess simulation error.

MCMC can be used to tackle famous problems, such as the travelling salesman problem. It is used to solve inverse problems in remote sensing. In image analysis, simple edge detection is often based on local gradient information only. MCMC can incorporate additional information such as smoothness, continuity and closure.

Credibility regions

One of the main appeals of MCMC is the use of large samples from the chain to form credible regions for posterior estimators, direct from the empirical distributions. This offers the possibility of assessing the variability of point estimates (Green 1995a; Besag, Green, Higdon, and Mengersen 1995). Also, we can use models that are believed to be most appropriate for the data, even if they require non-standard likelihoods and non-conjugate priors.

Sensitivity analysis

Sensitivity analysis is an important part of responsible statistical inference. Prior distributions can have a major influence on the posterior so the sensitivity of the posterior to the prior is an important issue. MCMC can be used for sensitivity analysis.

2.3.7 Problems with MCMC sampling algorithms

Spatial invariance

MCMC is a computational technique that requires a full probabilistic model. Sometimes the spatial invariance assumption is *not* justified strictly speaking; but the benefits that it offers may make us willing to accept it. This problem can be especially noticeable in biological microscopy because images may contain very heterogeneous image structures. For example, the z-axis may suffer from different blurring to that on the x-y plane. This is discussed in more detail in Chapter 5.

Convergence and efficiency of MCMC

There are convergence issues that need to be considered with MCMC.

The canonical example for illustrating the problems that can arise with single variable updating in MCMC is a mixture of two bivariate normal distributions, displaced diagonally from each other. As the distance between them increases, swaps between one mode and the other become increasingly rare. For a finite number of realisations of the Markov chain there may be little or no swaps but mathematically the Gibbs sampler and other MCMC algorithms are still irreducible. This is why MCMC may sometimes work badly.

Two methods for alleviating this are *multigrid* and *auxiliary* variables. For multigrid methods (Sokal 1989) the size of the sets of variables that are simultaneously updated is varied systematically from small to large and back again. The sets of variables are neighbouring pixels. In auxiliary variable methods (Swendsen and Wang 1987; Besag and Green 1993), the original variables x are augmented by additional variables u say, with $\pi_X(u \mid x)$ specified. Simulation relies on x and u being updated alternately, using a MCMC method such as Gibbs sampling. The effect is that the chain mixes more rapidly. The marginal for x is unchanged, therefore information extracted from it is valid.

Monitoring the rate of convergence is currently an active area of research (Rosenthal 1994; Roberts and Polson 1994). One approach is to monitor the output from the chain and use a function of the realisation to decide when convergence has occurred. Green (1995b) warns that it is difficult to draw inferences about parts of a distribution which have not been visited, without prior knowledge of the target distribution.

The large dimensionality of a state space may make monitoring of convergence difficult and the speed of convergence seems to decrease as the dimensionality increases (Green 1995a). This may be a problem with the 3D algorithm considered in Chapter 5.

Other problems

Phase transition problems within the prior models are another issue but this problem is offset by the fact that we are mainly concerned with the posterior distribution which is stabilised by the likelihood function.

MCMC must be designed to allow free movement around those regions of the model space that have substantial support in the posterior distribution.

We do not consider strategies for avoiding excessive autocorrelation or novel types of inference in MCMC algorithms.

2.3.8 Algorithms for finding the mode of a distribution

To help find the mode of the posterior distribution, we use two other algorithms: simulated annealing is stochastic and iterated conditional modes is deterministic.

Simulated annealing

Kirkpatrick et al. (1982) discovered an analogy between minimising the cost function of a combinatorial optimisation problem and the slow cooling of a solid. They called this algorithm *simulated annealing* as they were simulating the physical process of cooling a metal which is known as annealing. Simulated annealing is an MCMC, stochastic-optimisation algorithm for sampling from a target probability distribution. We use it in conjunction with the other MCMC algorithms, to find the image x with maximum probability in the posterior distribution $\pi_X(x) = f_{X|Y}(x, y)$.

Combinatorial optimisation involves attaching a real number to a finite or countably infinite number of states using a cost or *energy function.* The objective is to search the state space to find the state with the minimum energy. The modification, that enables simulated annealing to optimise rather than sample, is the raising of the target distribution to higher and higher powers over the course of the algorithm. This places an increasing probability at the globally optimum state. By using the energy in the Metropolis algorithm as the energy function in the simulated annealing algorithm and slowly reducing the temperature, Kirkpatrick et al. (1982) found a general solution to combinatorial optimisation problems.

An initial image is needed to initialise the algorithm.

Definition 2.15. The *maximum likelihood estimate* (MLE) is created by assigning to each pixel the colour which is closest to the record value for that pixel.

The resulting image is called the *closest mean classifier* (CMC).

Algorithm 2.4. SIMULATED ANNEALING.

- *Initialise to any convenient image, such as the closest mean classifier of the record.*
- *Simulate from a process with the distribution*

$$\pi_X^{(T)}(x) = \{\pi_X(x)\}^{1/T} \Big/ \sum_{x \in \Omega} \{\pi_X(x)\}^{1/T}. \tag{2.13}$$

  - *Choose an initial value for T. Complete a sweep of the image by sequentially updating each of the pixels in the image.*
  - *Remember the image with the lowest energy.*
  - *After each sweep of the image the parameter T is reduced, such that,*

$$T_t \geq T_{t+1} \qquad \text{and} \qquad \lim_{t \to \infty} T_t = 0,$$

    according to a predefined schedule.

- *The image with the lowest energy is the final answer.* □

T is called the temperature parameter. As $t \to 0$, our estimate of the true scene concentrates on the mode of the posterior probability.

Remark.

- The crux of the algorithm is that it might accept a move to a state with a higher energy function as a means of escaping from a local minimum.
- If the starting temperature is sufficiently high, the starting point is not critical because the algorithm can escape from local minima. Therefore for convenience, we use the maximum likelihood estimate to initialise the algorithm.
- The algorithm relies on a process of iterative improvement. This makes it easy to implement but it is slower than some 'greedy' algorithms, which can only converge to local minima.
- The algorithm has the potential to reach but *not to detect* the global maximum of the posterior distribution. In theory, the algorithm will find the mode of the posterior, if the temperature is decreased at a sufficiently slow rate.

Temperature schedules for simulated annealing

Because we are not sampling from a fixed distribution, the temperature must be decreased slowly (Geman and Geman 1984). In theory, the temperature parameter should be decreased on a log schedule but in practice we can only afford to decrease the temperature towards zero over a finite period of time. Consequently, we can only expect to find a local minimum. However, if the starting temperature is high enough, we expect to explore the state space sufficiently to find a good local minimum. The rate of decrease may also depend on the complexity of the energy function.

We consider the four families of temperature schedules proposed by Stander and Silverman (1994). These are shown in Table 2.1. This means that our schedule requires four parameters: starting and stopping temperatures, the desired number of sweeps of the image and the shape of the temperature schedule. This approach has the advantage that we can explicitly control the number of sweeps, the parameter to which the CPU time is most sensitive. All four temperatures were considered but the geometric temperature schedule is used throughout because it was found to work better in practice than other temperature schedules, not cooling too quickly or too slowly.

Example 2.3. TEMPERATURE SCHEDULES.
For example, suppose we wish to sweep an image using simulated annealing a finite number of times, say 50, and the temperature starts at 5 and drops towards 0. Then Figure 2-1 shows some examples from the families of temperature schedules listed in Table 2.1. In order to get from a temperature of 5 towards 0 in a finite number of sweeps, both the logarithmic and reciprocal temperatures drop rapidly towards zero during the first few sweeps. Thereafter, the algorithm makes few changes to the image and so is not exploring

Family of schedules	Temperature
Straight	$f + (l - f)(t - 1)/(T - 1)$
Geometric	$f(l/f)^{(t-1)/(T-1)}$
Reciprocal	$lf(T - 1)/\big(lT - f + (f - l)t\big)$
Logarithmic	$\frac{lf\left\{\log(T+1)-log(2)\right\}}{l\log(T+1)-f\log(2)+(f-l)\log(t+1)}$

Table 2.1: FAMILIES OF TEMPERATURE SCHEDULES. This table shows the four temperature schedules that were considered for simulated annealing: straight line, geometric, reciprocal and logarithmic. Each schedule has four parameters: T is the total number of sweeps, f is the starting temperature, l is the finishing temperature and t is the current sweep number.

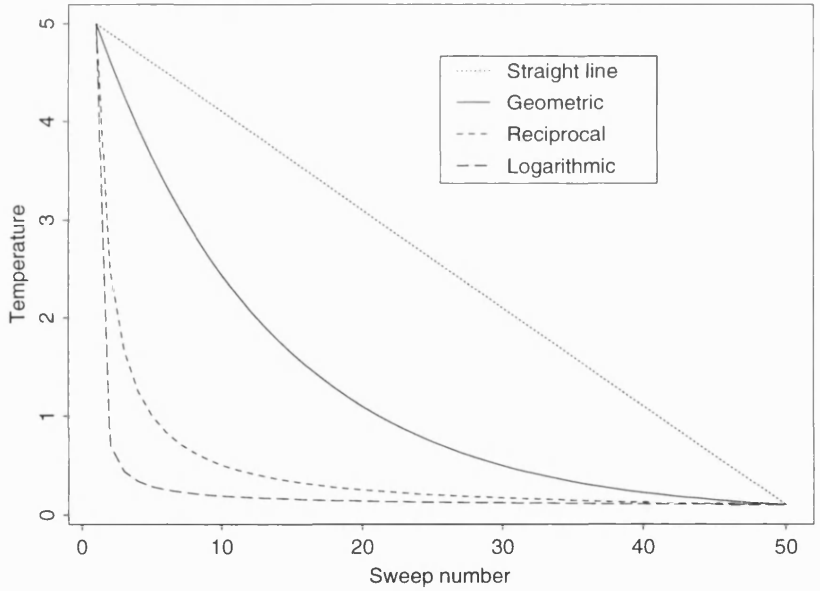

Figure 2-1: FAMILIES OF TEMPERATURE SCHEDULES. Some examples from the families of temperature schedules listed in Table 2.1 are shown here, where there are 50 sweeps with a starting and finishing temperature of 5 and 0.1, respectively. The logarithmic schedule was felt to drop too rapidly to explore the state space fully, as does the reciprocal schedule. On the other hand, the straight line was felt to decay too slowly. Thus the geometric schedule is preferred.

the state space as much as it does under the geometric schedule. The straight line schedule was found to decrease too slowly. Consequently the geometric schedule is favoured. □

Iterated conditional modes (ICM)

A deterministic, strictly downhill algorithm that Besag (1986) called *iterated conditional modes (ICM)* forces convergence to the 'nearest' local minimum from the starting image. We use it to find the image X with maximum probability in the distribution $\pi_X(x) = f_{X|Y}(x, y)$.

Algorithm 2.5. ITERATED CONDITIONAL MODES (ICM).

- *Start with the CMC of the record.*
- *For each pixel i in the image, update from x to x' where $x, x' \in \Omega$ and $x_{-i} = x'_{-i}$, by choosing*

$$x' = \max\{\pi_{X_i}(x''_i) : x''_{-i} = x_{-i}\},$$

  for $i = 1, \ldots, n$.
- *Repeatedly sweep the image until convergence.* □

ICM requires a good starting point which makes it sensitive to noise. ICM does not require a temperature schedule, as it merely focusses in on the nearest local minimum and convergence is guaranteed usually after only a few sweeps of the image. ICM is equivalent to running simulated annealing at a temperature of zero.

Comparisons between simulated annealing and ICM

Simulated annealing can achieve far lower energies than ICM but at a cost of more processing time. Simulated annealing sometimes accepts increases in the energy which allows this algorithm to escape from local minima. Simulated annealing also has the theoretical advantage that it will converge to the mode of the posterior (or any statistic of the posterior distribution) as the number of sweeps tends to infinity. In practice, only a finite number of sweeps are available and it is not clear how well the sampling mechanism performs at each temperature.

A well known problem can arise with simulated annealing if the target distribution exhibits multi-modality. For a fixed temperature, successive realisations become increasingly dependent as a local minimum is approached. Unless the algorithm can escape, the process may spend a long time restricted to that region. This problem is known as critical slowing down (Besag and Green 1992).

ICM is a strictly downhill or 'greedy' algorithm that converges deterministically to a local maximum of the posterior distribution. The algorithm cannot escape from a local minimum, so the final reconstruction depends heavily on the starting point. For example,

if the starting image has every pixel coloured white then the prior penalty in Equation (2.3) is zero. A pixel will change colour to escape this poor starting point only if the resulting increase in the prior penalty is more than offset by the corresponding reduction in the likelihood penalty. However, it is important to realise that it is the combination of a poor or naive starting point and the level of noise inherent in the image that results in a poor reconstruction from ICM. Consequently, given a sufficiently low level of noise, ICM can produce quite reasonable reconstructions even from poor starting images.

Combining simulated annealing and ICM

During the last few sweeps of the simulated annealing algorithm, the temperature drops close to zero. Ideally we would like to continue running the simulated annealing algorithm at a temperature of zero to converge on a local minimum (Geman, Geman, and Graffigne 1987). However, this would cause 'division by zero' errors in Equation (2.13). To ensure convergence, we use a combination of simulated annealing followed by a strictly downhill algorithm.

- Apply simulated annealing as per Algorithm 2.4.
- Initialise the downhill algorithm to the reconstruction which had the lowest energy from the simulated annealing algorithm, *à la* Stander and Silverman (1994).
- Apply the downhill algorithm until it converges.

We illustrate the benefit of this approach in §2.6.

2.4 Image estimators

2.4.1 Possible estimators

Following on from §2.2.5, the posterior distribution $f_{X|Y}(x,y)$ of Equation (2.6) expresses our knowledge of X given the prior model and the record Y. We can use the *empirical average* of l realisations to estimate the expectation of X,

$$E_{f_{X|Y}}(X) = \sum_{x\in\Omega} x f_{X|Y}(x,y) \approx \bar{x} \stackrel{\text{def}}{=} l^{-1} \sum_{t=1}^{l} x^{(t)}, \tag{2.14}$$

where $x^{(t)}$ is the state at time $t = 0, 1, 2, \ldots$. As will become clearer in later chapters, averaging realisations is not straightforward for the models used in this thesis, so we use point estimates of the posterior instead.

However, expectations present problems with averaging the realisations, as will become clearer in later chapters.

The aim is to sample from the posterior of our distribution in order to arrive at a point estimate for the true image. There are two commonly used point estimates.

Definition 2.16. The maximum a posteriori estimate (MAP) is the point estimate of the image $x \in \Omega$ that maximises the posterior distribution $f_{X|Y}(x, y)$.

Therefore we want to find an image X, in the predefined set of allowable images Ω, that maximises $f_{X|Y}$.

Definition 2.17. The marginal posterior mode estimate (MPM) is the point estimate of the image that maximises the posterior probability $f_{X_i|Y}(x_i, y)$, for $i = 1, \ldots, n$ (Marroquin, Mitter, and Poggio 1987).

MPM is equivalent to minimising a loss function which incurs a penalty of one for each misclassified pixel. In an imaging context, MPM can be implemented by sampling from $f_{X|Y}$ and choosing the most frequently occurring colour at pixel X_i as the estimate for the colour of that pixel.

2.4.2 Comparisons between estimators

The point estimates MAP and MPM are not necessarily equal. Greig, Porteous, and Scheult (1989) contains examples of MAP estimators which have high penalties under the MPM loss function.

Which estimator is more suitable depends on the application. For example, MPM may be better if we are interested in a statistic of the number of pixels of a particular colour, such as the classification of land from ariel images. MAP may be better at restoring groups of pixels such as in shape analysis (Ripley 1988).

The MAP estimate is a convenient choice for our subpixel problem and it is this estimate that we pursue. That is, we seek the mode of the posterior distribution, the MAP estimate,

$$\hat{X} = \arg\max_{x \in \Omega} \{f_{X|Y}(x, y)\}.$$

For simplicity, denote $\hat{X}$ by X. The reasons why this estimator is more convenient will become clear in later chapters.

2.5 Measuring the quality of a reconstruction

There are various measures of how good a reconstruction is. The most common measure is simply a visual inspection of the final image. If it contains no obvious discrepancies then it could be said to have passed a visual inspection. However, some quantitative measures that are clear and objective are essential.

A commonly used measure of 'goodness-of-fit' is the energy of the image, defined in Equation (2.7). This value is readily available and objective. For ICM, the energy decreases monotonically, while for simulated annealing it generally decreases as the temperature drops towards zero in Equation (2.13).

Another measure of discrepancy between two images is to count the number of pixels in one of the images where the corresponding pixel in the other image is a different colour. This is particularly convenient for binary images. In simulated examples, the true image is known so we can count discrepancies between a reconstruction and the true image.

For simulated annealing, we acknowledge that it is not sufficient to show comparisons based on one realisation from the chain, because it is a stochastic algorithm. The reconstructions will vary based on the samples drawn from the posterior distribution at each update. So any statistics based on a single chain must be used with caution.

2.6 An application of some of the MCMC algorithms

To illustrate simulated annealing and ICM, we show an example where we reconstruct a binary image from the state space of all binary images whose size equals that of the record. So $\Omega \stackrel{\text{def}}{=} \{x_i \in \{b, w\} : i = 1, \ldots, n\}$, where b and w represent black and white coloured pixels, respectively. The values attributed to the colours in this example are $b = 1$ and $w = 0$. We run a Markov chain on the posterior distribution so $\pi = f_{X|Y}$. A first-order neighbourhood is used. The energy arising from the prior, in Equation (2.3), is the number of discrepant first order neighbours multiplied by a smoothing penalty, $\beta = 1$. So

$$U_X(x) \stackrel{\text{def}}{=} \frac{\beta}{2} \sum_{i=1}^{n} \sum_{j \in \delta_i^1} I_{[x_i \neq x_j]}, \quad \text{where } I_{[x_i \neq x_j]} = \begin{cases} 1 & \text{if } x_i \neq x_j, \\ 0 & \text{if } x_i = x_j. \end{cases} \tag{2.15}$$

The factor of $\frac{1}{2}$ is necessary to avoid double counting. The record $Y = (y_1, y_2, \ldots, y_n)$ is assumed to be derived from the true image by the pixelwise addition of independent, Gaussian noise with known variance. So for this example, the degradation model is

$$Y_i \stackrel{\text{def}}{=} h_i(x) + e_i, \qquad \text{where } e_i \sim N(0, 0.5).$$

Each pixel is of single colour and for simplicity, there is no blurring in this example. From Equation (2.5), the function which measures the mixtures of colours in the reconstruction is

$$h_i(x) \stackrel{\text{def}}{=} \begin{cases} b & \text{if } x_i \text{ is coloured black,} \\ w & \text{if } x_i \text{ is coloured white.} \end{cases}$$

On substituting these values into Equation (2.6), the posterior function becomes

$$f_{X|Y}(x, y) \quad \propto \quad \exp\left\{ -\frac{\beta}{2} \sum_{i=1}^{n} \sum_{j \in \delta_i^1} I_{[x_i \neq x_j]} - (2 \times 0.5)^{-1} \sum_{i=1}^{n} \big(y_i - h_i(x)\big)^2 \right\}, \qquad x \in \Omega.$$

Plot (a) in Figure 2-2 shows the true 64×64 scene. The record is shown in Plot (b) as a grey scale image, after noise with distribution $N(0, 0.5)$ is artificially added. Note that

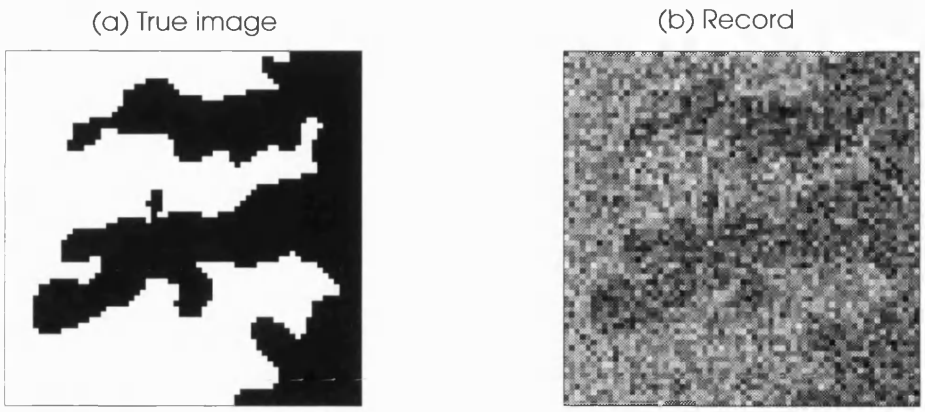

Figure 2-2: SIMULATED ANNEALING AND ICM — 1/3. The 64 × 64 pixel, binary image on the left is the true scene. Artificial noise is added and the resulting record is shown on the right as a grey scale image. Note that the colouring scales in the two images are different.

the colours in the image of the record have been scaled differently to those in the other images, so direct comparisons are not possible. In Plot (a), black and white represent the foreground and background colours, respectively; in Plot (b) they represent the greatest and least record value, respectively.

The simplest estimate of the true image is the CMC (see §2.3.8 on page 34). The result is shown in Plot (a) of Figure 2-3. This estimate is too crude to be of any real value but it can be used as a simple and convenient starting point for simulated annealing. Note however that any starting point, such as an all white image, would also work with simulated annealing provided that a sufficiently high starting temperature is used and the algorithm is run for a sufficient period.

A Markov chain, using the Gibbs sampler, is implemented using *single-site updating*. This means that only one pixel is updated at a time. The simulated annealing algorithm is based on a geometric schedule with starting and stopping temperatures of 5 and 0.1, respectively. The number of sweeps used is 1,200.

Plots (b)—(f) of Figure 2-3 show the reconstructions after $200, 400, \ldots, 1,000$ sweeps respectively. For each reconstruction the energy is printed. It shows a steadily decreasing value as is to be expected. Other comparisons are also shown. The number of times corresponding pixels differ in pairs of images is an easy and intuitive way to compare reconstructions. For each image in Plots (b)—(f), the number of disagreements between corresponding pixels in that image and the true image (true), the CMC image (cmc) and the previous reconstruction (prev) are shown. For example, when comparing Plot (b) and Plot (c) there are 1,433 corresponding pixels coloured differently. As the temperature decreases, the number of differences between the reconstruction and the true image decreases. So does the number of differences with the previously shown reconstruction because the algorithm makes relatively few changes after the first few hundred sweeps.

Figure 2-4(a) shows the final image after 1,200 sweeps using simulated annealing. This

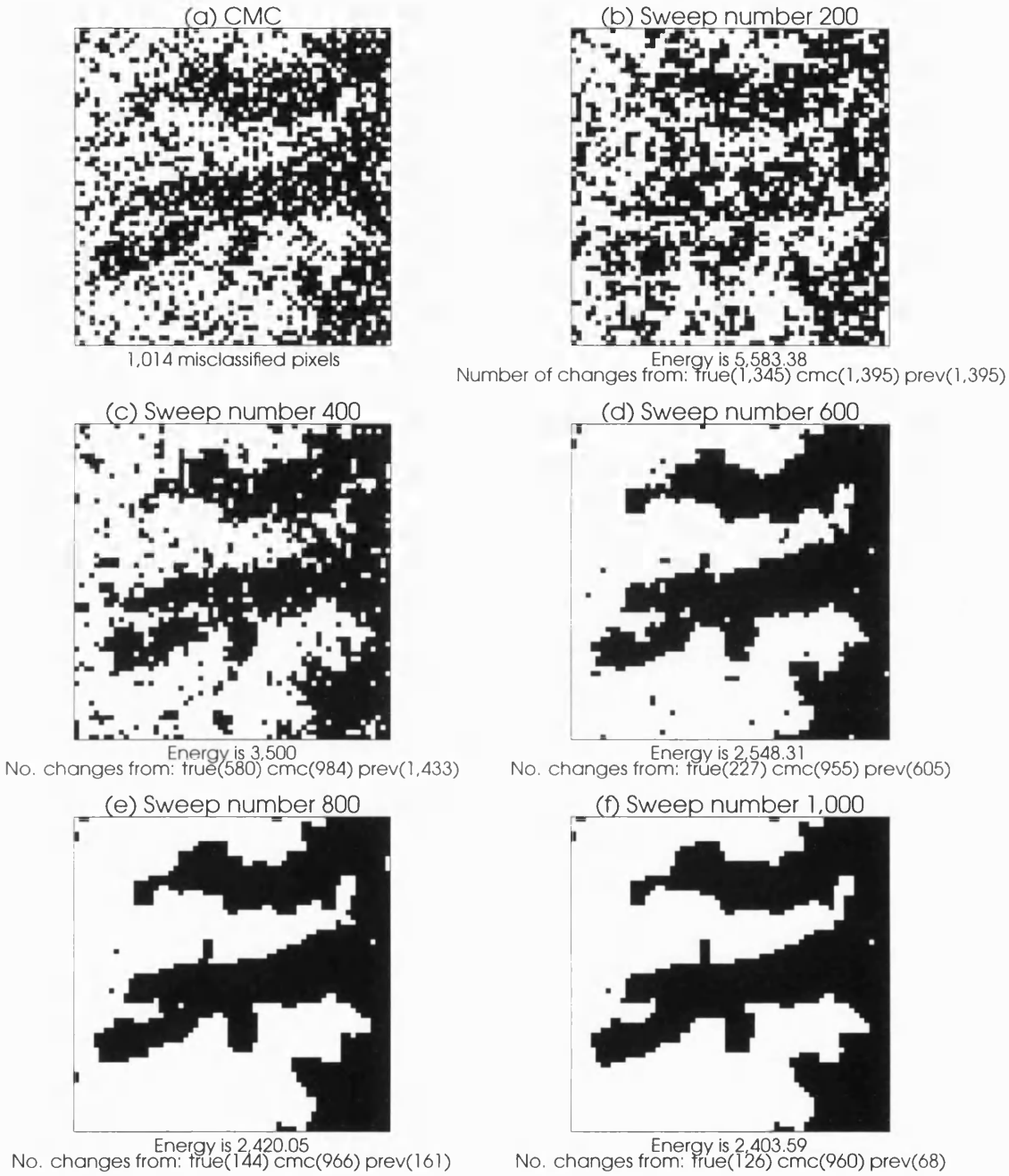

Figure 2-3: SIMULATED ANNEALING AND ICM — 2/3. This figure shows the reconstruction from the simulated annealing algorithm on the image in Figure 2-2 (a), using the CMC as the starting point (Plot (a)) and following a geometric schedule. Reconstructions are shown after every 200 sweeps of the image, in Plots (b)—(f). The subtitle in each plot shows the energy for each image and the number of misclassified pixels between the current plot and the true reconstruction (Figure 2-2 (a)), the CMC reconstruction (Plot (a)) and the previous reconstruction.

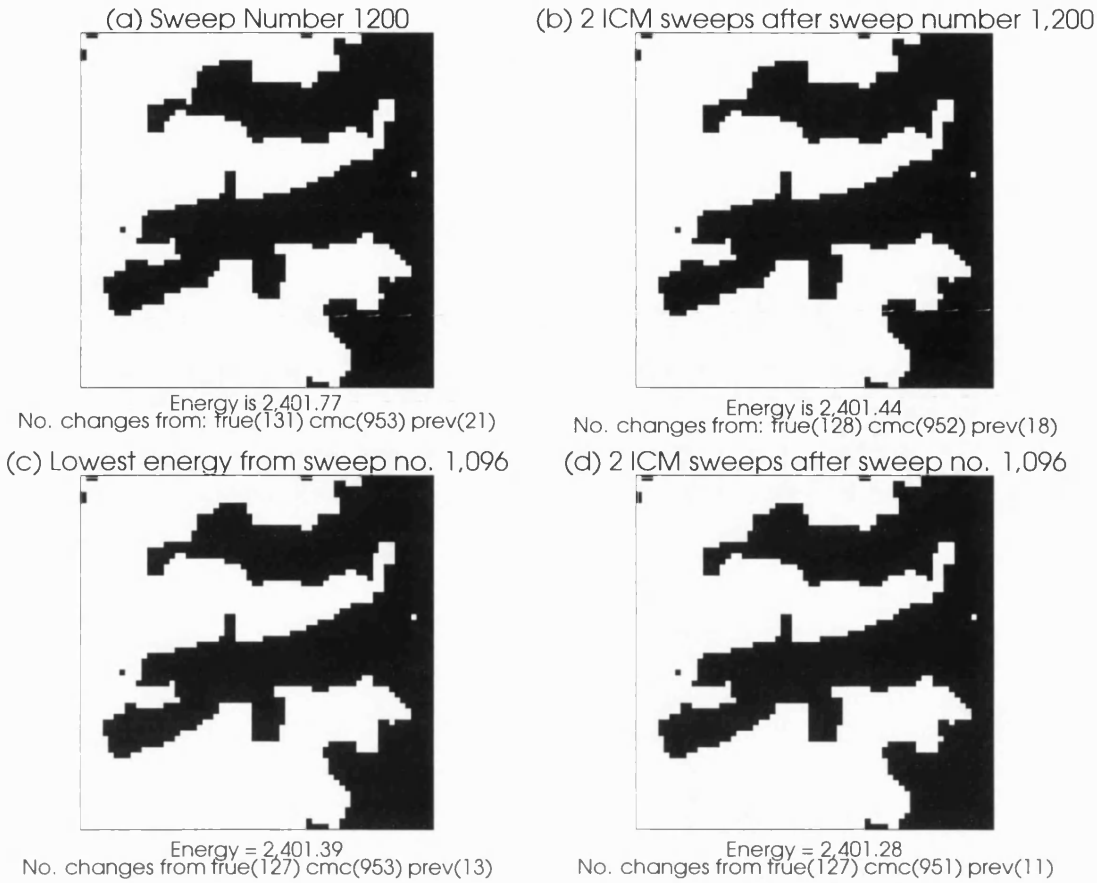

Figure 2-4: SIMULATED ANNEALING AND ICM — 3/3. Plot (a) shows the reconstruction after 1,200 sweeps using simulated annealing. Plot (b) shows the reconstruction after applying ICM to convergence (2 sweeps) on the image in Plot (a). Plot (c) shows the reconstruction with the lowest energy from the simulated annealing stage, which was sweep number 1,096. Plot (d) shows the reconstruction after applying ICM to convergence on sweep number 1,096.

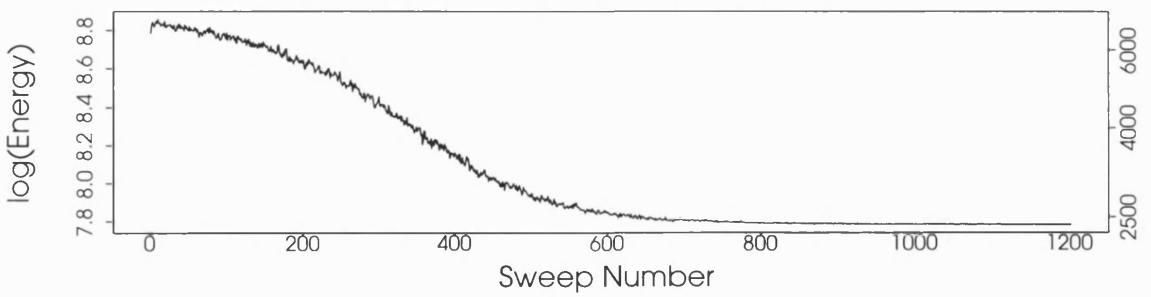

Figure 2-5: LOG OF THE ENERGY VERSES THE SWEEP NUMBER. The log of the energy for each reconstruction is plotted against the sweep number. The original energy scale is shown on the right. The energy is not monotonically decreasing under simulated annealing.

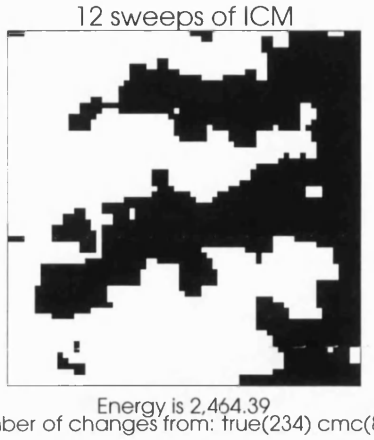

Figure 2-6: ICM ONLY. This figure shows the reconstruction from applying ICM, *without* first using simulated annealing. It took only 12 sweeps for the algorithm to converge and the energy is fairly low. However, there are a significant number of misclassified pixels compared to both the true and CMC images. This shows the benefit of applying the simulated annealing algorithm before applying the ICM algorithm.

image has an energy of 2,401.77 using Equation (2.6). Applying ICM to convergence requires two further sweeps of the image and results in an image energy of 2,401.44, shown in Figure 2-4(b). This suggests that simulated annealing had almost settled into the nearest local minimum. The reconstruction which achieved the lowest energy is shown in Figure 2-4(c). This occurred at sweep number 1,096, with an energy of 2,401.39. Applying ICM to convergence again requires a further 2 sweeps, giving a final estimate with an energy of 2,401.28. Again, ICM was of little help because the difference between the energies in Plots (c) and (d) is trivial. Usually, we expect Plot (d) to have lower energy than Plot (b) because Plot (d) starts the ICM stage using a reconstruction with a lower energy. In this case, the two energies are almost the same. Plots (b) and (d) also show that the number of misclassified pixels compared with the true image in each case is 128 and 127, respectively. Normally Plot (d) does much better than Plot (b), both in terms of the energy function and the number of misclassified pixels. A contributing factor in this case is that the relatively large number of simulated annealing sweeps ensures that the chain has converged to the mode of the posterior.

The log of the energy for each reconstruction is plotted against the sweep number in Figure 2-5. The energy is not monotonically decreasing under simulated annealing. This reflects the fact that the algorithm can escape from a local minimum by accepting proposals that have a higher energy, with a certain probability. However, ICM is monotonically decreasing. When ICM is applied there is usually a sudden drop in energy, due to ICM removing isolated pixels that simulated annealing has accepted. Thereafter, the energy continues to drop under ICM but at a slower rate. In this case, the simulated annealing stage of the algorithm is very close to a local minimum so ICM has little effect. Although the number of sweeps required for ICM is not known in advance, it is usually very small, relative to simulated annealing.

Figure 2-6 shows the reconstruction from applying ICM, without first using simulated annealing. It took only 12 sweeps for the algorithm to converge and the energy is fairly low. However, there are a significant number of misclassified pixels compared to both the

true and CMC images. Compared to the true image, simulated annealing plus ICM had 127 misclassifications compared to 234 for ICM on its own. ICM suffers because it was given a poor starting point. It is for this reason that we first apply simulated annealing before applying ICM to the simulated annealing reconstruction with the lowest energy, in subsequent chapters.

2.7 Summary of chapter

The purpose of this chapter is to define some notation and introduce the basic statistical models and computational algorithms that are used throughout the remaining chapters. To achieve this:

- We define some basic assumptions and notation.
- The prior, likelihood and posterior distributions of the Bayesian model are introduced.
- We outline standard algorithms for searching over a high-dimensional density, namely:
  - the Gibbs sampler,
  - the Metropolis algorithm and
  - the Hastings algorithm.

  Comparisons are made between them, characteristic features are outlined and their relative advantages and disadvantages are discussed.
- Two algorithms for optimising a function are used, namely:
  - simulated annealing and
  - iterated conditional modes.

  The two algorithms are contrasted. Problems associated with the temperature schedule for simulated annealing are briefly mentioned.
- Based on the posterior density, various estimates of the true image are discussed, namely:
  - the maximum a posteriori and
  - the marginal posterior modes estimators.
- Various ways of measuring the quality of reconstruction are stated.
- To illustrate the algorithms at the full pixel level, a typical example is analysed using the Gibbs sampler, simulated annealing and ICM.

Chapter 3

Discrete 2D subpixel reconstruction

By small sample,
we may judge the whole piece.
Miquel le Cervantes

It would be as useless to perceive
how things 'actually look'
as it would be to watch the random dots
on un-tuned television screens.
Marvin Lee Minsky

3.1 Introduction

A subtle use of partial conditioning in high-dimensional, multi-modal applications occurs when we use the posterior distribution at different scales. For image analysis, these scales are associated with a coarsening or refinement of a pixel lattice. In this chapter, a discrete approach to subpixel reconstruction is advocated. The basic idea is to extend the full pixel model outlined in the previous chapter to allow each pixel to contain two regions of colour.

The size and orientation of the pixel grid used to represent the image are independent of the continuous, true image so it is rarely the case that all pixels on this grid are wholly one colour. Instead, the pixels around the edge of an object will contain more than one colour. An intuitive and simple way to model this feature is to subdivide each pixel in the original image into a $2^m \times 2^m$ two-dimensional array of subpixels, in order to more closely match the proportion of the two colours inside each pixel.

A naive subdivision of each pixel very quickly leads to too many combinations and excessive computation. For example, if all subpixels within a given pixel are updated *simultaneously* then it is computationally prohibitive to go straight to the m^{th} subpixel level. Wolberg and Pavlidis (1985) show how the number of possible combinations can be reduced to reflect the objectives of the reconstruction. They do this by dealing with 3×3 windows and 5×5 windows using a detailed but complicated set of rules to reduce the number of combinations. However, Wolberg and Pavlidis (1985) do not consider subpixel models.

We adopt an alternative approach by proceeding in steps. Each pixel, and subsequently each subpixel, is repeatedly divided into a 2×2 subwindow. Thus the subpixel image at the m^{th} level is created by cascading through images with finer and finer resolution, until each pixel consists of a $2^m \times 2^m$ array of subpixels. The problem is broken into systematic and consistent stages so that the algorithm becomes computationally feasible. This procedure is often called *cascading*.

Example 3.1. EXAMPLE OF A SUBPIXEL CASCADE.
In Figure 3-1, the original reconstruction is a 2×2 pixel image at level $L^{(0)}$. This becomes a

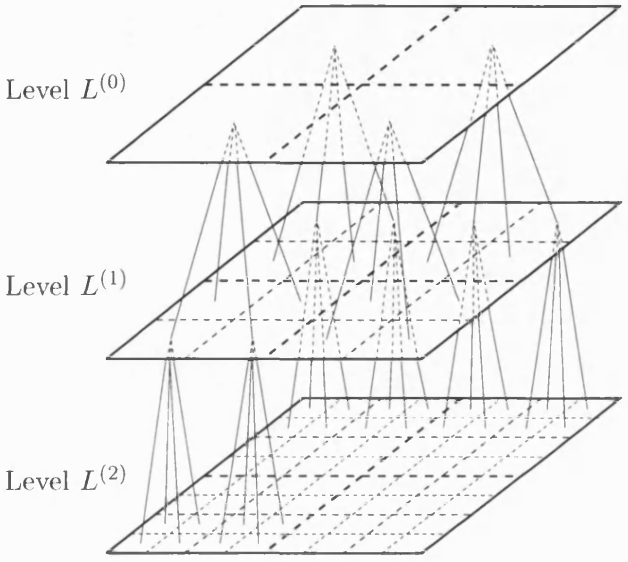

Figure 3-1: EXAMPLE OF A SUBPIXEL CASCADE. Three different levels of the cascade are shown. To get from one level to the next, each pixel is subdivided into 2×2 blocks of subpixels.

4×4 and an 8×8 pixel image at levels $L^{(1)}$ and $L^{(2)}$ respectively, by repeatedly subdividing each pixel into 2×2 subpixels. □

The motivation for the subpixel model is based on Jubb and Jennison's (1991) cascade algorithm, which we refer to as the 'superpixel' algorithm, to distinguish it from the subpixel model discussed here. A brief summary is given in §3.5.

3.2 The model

3.2.1 Some notation

The image that we seek is on a much finer lattice than that of the record $Y = (y_1, y_2, \ldots, y_n)$. We denote this image by $X = X^{(M)} = \{X_1^{(M)}, \ldots, X_n^{(M)}\}$ where $X_i^{(M)} = \{X_{ij}^{(M)} : j = 1, \ldots, 2^{2M}\}$, for $i = 1, \ldots, n$, and realisations are denoted by $x_i^{(M)} = \{x_{ij}^{(M)} : j = 1, \ldots, 2^{2M}\}$.

Definition 3.1. The sample space or *image space* is $\Omega = \Omega^{(M)} = \{x_i^{(M)} : x_{ij}^{(M)} \in \{b, w\}; i = 1, \ldots, n; j = 1, \ldots, 2^{2M}\}$, where b and w represent black and white coloured subpixels, respectively.

So Ω and X lie on an $n \times 2^M \times 2^M$ lattice, where M is predefined and fixed during the reconstruction. The *level* of the cascade is denoted by L and indexed by $m = 0, 1, \ldots, M$. Within the m^{th} level $L^{(m)}$, each full pixel $X_i^{(m)}$ corresponds to a single record value Y_i and $X_i^{(m)}$ is subdivided into a $2^m \times 2^m$ block of subpixels. Note that $i = 1, 2, \ldots, n$ is used to index the blocks of subpixels and $j = 1, 2, \ldots, 2^{2m}$ indexes the subpixels within each block, in the image X.

3.2.2 The prior distribution

A first and second order neighbourhood is used for the prior. From the prior in Equation (2.3), the energy is the number of discrepant neighbours multiplied by a smoothing penalty. So the prior distribution is

$$\begin{aligned} f_X(x) &\propto \exp\{-U_X(x)\} \\ &\stackrel{\text{def}}{=} \exp\left\{-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{2^{2M}}\left\{\beta_1 \sum_{k,l\in\delta_{ij}^1} I_{[x_{ij}\neq x_{kl}]} + \beta_2 \sum_{k,l\in\delta_{ij}^2} I_{[x_{ij}\neq x_{kl}]}\right\}\right\}, \end{aligned} \tag{3.1}$$

where $x \in \Omega$, I is the indicator function defined in Equation (2.15) and β_1 and β_2 are the smoothing penalties for the first and second order neighbourhoods, respectively. From the notation in §2.2.3, δ_{ij}^1 and δ_{ij}^2 denote the first and second order neighbourhoods for the subpixel X_{ij}. The first and second order interaction terms in the prior model should reflect the prior belief in the MRF. The models in Chapters 4 and 5 rely on first order neighbours only but in this chapter we include the second order interactions.

3.2.3 The likelihood function

As in the last chapter, the record Y is assumed to be derived from the true image by the pixelwise addition of independent, Gaussian noise with known variance (see Equation (2.4) on page 23). There is no blurring in this model for simplicity. So, the likelihood model follows Equation (2.4). The expected value of each record element is

$$E(Y_i) = h_i(x) = 2^{-2M}\sum_{j=1}^{2^{2M}} x_{ij}^{(M)}, \quad \text{where } x_{ij}^{(M)} = \begin{cases} b & \text{if } x_{ij}^{(M)} \text{ is coloured black,} \\ w & \text{if } x_{ij}^{(M)} \text{ is coloured white,} \end{cases}$$

for $i = 1, \ldots, n$. The factor 2^{-2M} arises from each subpixel $x_{ij}^{(M)}$ occupying 2^{-2M} of the block of subpixels $x_{i\cdot}$ corresponding to each y_i. Putting this in Equation (2.5) gives

$$f_{Y|X}(x, y) \propto \exp\{-(2\sigma^2)^{-1}\|y - h(x)\|^2\} = \exp\left\{-(2\sigma^2)^{-1}\sum_{i=1}^{n}\big(y_i - h_i(x)\big)^2\right\}. \tag{3.2}$$

3.2.4 The posterior distribution

From Equation (2.6), the posterior distribution becomes

$$
\begin{aligned}
f_{X|Y}(x,y) &\propto \phi(x) \\
&\stackrel{\text{def}}{=} \exp\Bigg\{-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{2^{2M}}\Big\{\beta_1 \sum_{k,l\in\delta_{ij}^1} I_{[x_{ij}\neq x_{kl}]} + \beta_2 \sum_{k,l\in\delta_{ij}^2} I_{[x_{ij}\neq x_{kl}]}\Big\} \\
&\qquad - (2\sigma^2)^{-1}\sum_{i=1}^{n}\big(y_i - h_i(x)\big)^2\Bigg\},
\end{aligned}
\tag{3.3}
$$

for $x \in \Omega$. The function $\phi(x)$ is the posterior distribution up to normality. We now have a model to do the reconstruction at the finest level of resolution, $L^{(M)}$.

3.3 The algorithm

3.3.1 An overview

In this section, we describe the subpixel cascade algorithm. It is a method for maximising the function $\phi(x)$ in Equation (3.3), over $x \in \Omega \equiv \Omega^{(M)}$. The strategy is to maximise $\phi(x)$ over a sequence of subsets of Ω, where values of $x_{ij}^{(M)}$ are constrained to be constant over $2^{M-m} \times 2^{M-m}$ blocks, which partition $X^{(M)}$ at level $L^{(m)}$. Call this image space $\Omega^{(m)}$. The solution at level $L^{(m)}$ will then serve as the starting point for optimisation at level $L^{(m+1)}$.

At level $L^{(m)}$, where $m < M$, we are still working with the reconstruction $X^{(M)}$ but there are constraints on the values that $X^{(M)}$ is allowed to take. We introduce $X^{(m)}$ as a shorthand for a value of $X^{(M)}$ satisfying these constraints, without all the repeated values of elements that are set equal to each other by constraint. At level $L^{(m)}$, $X^{(m)} = \{X_1^{(m)}, \ldots, X_n^{(m)}\}$ where $X_i^{(m)} = \{X_{ij}^{(m)} : j = 1, \ldots, 2^{2m}\}$ for $i = 1, \ldots, n$. It takes values $x_i^{(m)} = \{x_{ij}^{(m)} : j = 1, \ldots, 2^{2m}\}$. The sample space is $\Omega^{(m)} = \{x_i^{(m)} : x_{ij}^{(m)} \in \{b, w\}; i = 1, \ldots, n; j = 1, \ldots, 2^{2m}\}$. Strictly speaking, this definition of $\Omega^{(m)}$ differs from that given in the previous paragraph. In the previous definition $x^{(m)}$ contains $n \times 2^{2M}$ elements, with constraints on the values that they can take; here each $x^{(m)}$ contains only $n \times 2^{2m}$ elements without constraints. However, we use $\Omega^{(m)}$ to refer to both spaces.

$X^{(m)}$ is related to $X^{(M)}$ in a way which it is easy to understand but awkward to write out formally. Basically, each $X_i^{(M)}$ contains 2^{2M-2m} copies of each $X_{ij}^{(m)}$, where $j = 1, \ldots, 2^{2m}$ in the appropriate order. We can also think of $X^{(m)}$ as an image with $n \times 2^{2m}$ pixels.

3.3.2 Ensuring a consistent model while cascading

At level $L^{(m)}$, our aim is to maximise $\phi(x)$ over $x \in \Omega^{(m)}$. This is equivalent to maximising $\phi(x^{(m)})$ over $x^{(m)} \in \Omega^{(m)}$. We show how this problem arises for each m as a MAP estimation problem for a certain prior on $x^{(m)} \in \Omega^{(m)}$ and a likelihood function. For

each $m = 0, \ldots, M$, the function $\phi(x)$ is not itself a p.d.f. for $x \in \Omega^{(m)}$. It needs to be normalised and the normalising constant varies with m. This is not a constraint in practice because the essence of the MCMC algorithms in Chapter 2 is that they do not need this normalising constant.

To clarify the algorithm, it is helpful to think about a 'model' at each level of the cascade. We define these models because the processing at each level $L^{(m)}$, where $m = 1, \ldots, M-1$, has a prior, likelihood and posterior distribution that correspond to the distribution at level $L^{(M)}$. The models at different levels are automatically consistent as they all come from the single original model at level $L^{(M)}$, defined in §3.2. This contrasts with Jubb and Jennison (1991), who worked with non-consistent models at each level in the superpixel process but still hoped that this would find a good estimate of the MAP estimate at the bottom, full pixel level. See §3.5 on page 59 for a summary.

In order to produce a reconstruction at the M^{th} level, we start at the level of the record, level $L^{(0)}$, and proceed in steps. Each pixel, and subsequently each subpixel, is repeatedly divided into a 2×2 subwindow until each record value has a corresponding block in the reconstruction of size $2^M \times 2^M$. While doing this, it is essential to ensure consistency between the parameters in the model at different levels in the cascade.

The variance of the image at each level is unchanged, so each element of the record Y_i must be compared with the average colouring of the corresponding block of subpixels $X_i^{(m)}$, for consistency. In addition, we would like to compare the energy of the reconstructions at different levels. So some standardisation of the prior penalty is needed because the number of subpixel blocks at each level is different. In moving from coarse-to-fine resolution, the number of subpixel blocks within each $X_i^{(m)}$ increases by a factor of four between levels $L^{(m)}$ and $L^{(m+1)}$. So $4|X^{(m)}| = |X^{(m+1)}|$, where $m = 0, \ldots, M-1$. Also the first level $L^{(0)}$ has one pixel corresponding to each of the n record elements. So $|X^{(0)}| = n$.

Ensuring the prior distribution is consistent

If the MRF models at each level in the cascade are to be consistent, the value of the interaction strength between neighbouring pixels needs to be reduced as the cascade proceeds through each level, from $L^{(0)}$ to $L^{(M)}$. This would allow the prior energy function for each level in the cascade sequence to be compared on a consistent basis.

The approach used is to consider the reconstruction $X^{(m)}$ at the finest resolution, when $m = M$. For this level, each element of the record Y_i has a corresponding block of $2^M \times 2^M$ pixels in the final reconstruction, $X^{(M)}$. The earlier, coarser reconstructions in the algorithm are also treated as consisting of blocks of subpixels, of size $2^m \times 2^m$, for $m = 0, 1, \ldots, M-1$, but now the pixels are constrained to be the same colour within each block of subpixels. For example, at the m^{th} level, each Y_i has a corresponding block of $2^m \times 2^m$ subpixels which can be broken down into 2×2 non-overlapping sub-blocks, each of size $2^{m-1} \times 2^{m-1}$. The only way that the fine-resolution reconstruction $X^{(m)}$ can correspond to the coarser-level reconstruction $X^{(m-1)}$ is if the 2×2 block of subpixels in the fine-resolution reconstruction $X^{(m)}$ are constrained to be all the same

colour. Consequently, the energy of the reconstruction is the same immediately before and after the split.

Definition 3.2. At the m^{th} level, define $\beta_1^{(m)}$ and $\beta_2^{(m)}$ to be the *first* and *second* order interaction terms in the prior model, respectively.

At the finest level of resolution, $m = M$, we define $\beta_1^{(M)} \stackrel{\text{def}}{=} \beta_1$ and $\beta_2^{(M)} \stackrel{\text{def}}{=} \beta_2$, where β_1 and β_2 are data dependent parameters chosen at the outset. We now need to specify the correspondence between the prior penalties at earlier levels in the cascade. To explain how these relations are derived, consider the following example.

Example 3.2. ENSURING A CONSISTENT PRIOR DURING THE CASCADE.
Suppose we choose $M = 2$ for the final reconstruction and consider the interaction between

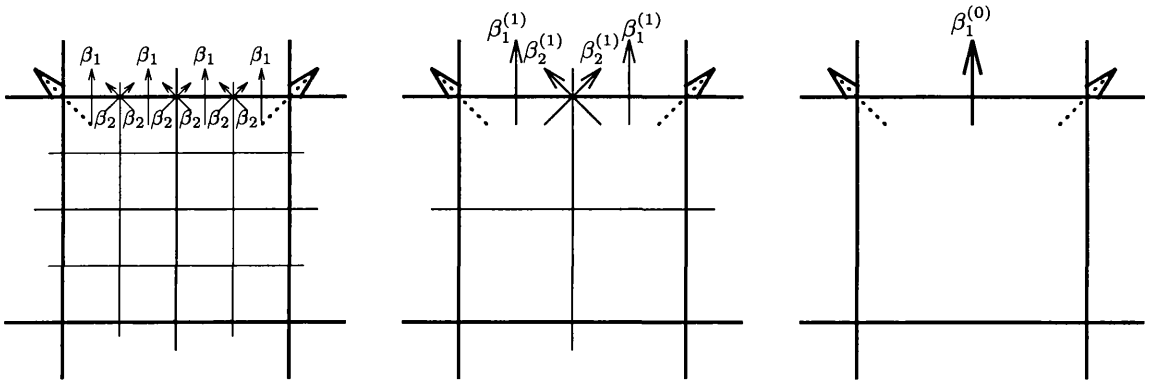

Figure 3-2: ENSURING A CONSISTENT PRIOR DURING THE CASCADE. A pixel at the original level $L^{(0)}$ is split into a block of 2 × 2 subpixels at level $L^{(1)}$ and into 4 × 4 subpixels at level $L^{(2)}$. The interactions between this block of pixels and the corresponding block of pixels immediately to the north are indicated by arrows ↑. If the effect of the interaction parameters at each level is to be consistent then the sum of the penalties arising from the arrows at each level should be the same.

a single record element and its neighbours to the north. At the finest level $L^{(2)}$, each record element is reconstructed using a 4 × 4 block of pixels. Figure 3.2 illustrates that the interaction with the corresponding block of pixels in the record element to the north is $4\beta_1^{(2)} + 6\beta_2^{(2)} = 4\beta_1 + 6\beta_2$. The corresponding interaction terms at levels $L^{(1)}$ and $L^{(0)}$ are $2\beta_1^{(1)} + 2\beta_2^{(1)}$ and $\beta_1^{(0)}$, as shown in Figure 3.2. Notice that the interaction with the block of pixels to the north-east and north-west $\beta_2^{(m)}$ (dotted arrows) is the same from one level of the cascade to the next. We need to choose values for $\beta_1^{(0)}$ and $\beta_1^{(1)}$ to ensure that the interaction with the pixel to the north is the same regardless of the level of the cascade. □

To ensure the prior penalties at successive levels in the cascade are comparable, the

following two relations are used:

$$\beta_2^{(m)} \stackrel{\text{def}}{=} \beta_2 \tag{3.4a}$$

$$\beta_1^{(m)} \stackrel{\text{def}}{=} [2^M\beta_1 + 2\beta_2(2^M - 2^m)]/2^m, \quad \text{for } m = 0,1,2,\ldots,M. \tag{3.4b}$$

We can now specify the 'effective' prior distribution for the image at the m^{th} level in the cascade as

$$f_X^{(m)}(x^{(m)}) \propto \exp\left\{-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{2^{2m}}\left\{\beta_1^{(m)}\sum_{k,l\in\delta_{ij}^1} I_{[x_{ij}^{(m)}\neq x_{kl}^{(m)}]} + \beta_2^{(m)}\sum_{k,l\in\delta_{ij}^2} I_{[x_{ij}^{(m)}\neq x_{kl}^{(m)}]}\right\}\right\}, \tag{3.5}$$

where $x^{(m)} \in \Omega^{(m)}$, δ_{ij}^1 and δ_{ij}^2 are the sets of first and second order neighbourhoods for $x_{ij}^{(m)}$, respectively. At $m = M$, Equation (3.5) is the same as Equation (3.1).

Ensuring the likelihood distribution is consistent

At the first level $L^{(0)}$, each element of the record Y_i has a corresponding pixel $X_i^{(0)}$ in the reconstruction. As the reconstruction moves from the original coarse resolution, at which the record was observed, to a finer resolution, the number of subpixels corresponding to each record increases. At the m^{th} level in the algorithm, each element in the record Y_i has a corresponding block of $2^m \times 2^m$ subpixels in the reconstruction $X^{(m)}$. This allows the mean of each record to be reconstructed as a mixture of regions of two different colours.

Jubb and Jennison (1991) originally used the cascade algorithm to aggregate records. This has the advantage of increasing the signal-to-noise ratio (SNR). However for subpixel resolution, the records are not aggregated so there is no increase in the SNR. Instead, to calculate the likelihood penalty for the image at the m^{th} level of the cascade, we compare each element of the record y_i with the average value of the colours in the corresponding block of $2^m \times 2^m$ subpixels, $h_i^{(m)}(x)$. The average value, from the mixture of the two colours, in the block of subpixels $x_i^{(m)}$ is

$$E(Y_i) = h_i(x^{(m)}) \stackrel{\text{def}}{=} 2^{-2m}\sum_{j=1}^{2^{2m}} x_{ij}^{(m)}, \quad \text{where } x_{ij}^{(m)} = \begin{cases} b & \text{if } x_{ij}^{(m)} \text{ is coloured black,} \\ w & \text{if } x_{ij}^{(m)} \text{ is coloured white,} \end{cases}$$

and j is summed over the subpixels lying within the i^{th} block of subpixels in $X^{(m)}$. So the 'effective' likelihood distribution for the the m^{th} level in the cascade is

$$\begin{aligned} f_{Y|X}^{(m)}(x^{(m)}, y) &\propto \exp\{-(2\sigma^2)^{-1}\|y - h(x^{(m)})\|^2\} \\ &= \exp\left\{-(2\sigma^2)^{-1}\sum_{i=1}^{n}\left(y_i - h_i(x^{(m)})\right)^2\right\}. \end{aligned} \tag{3.6}$$

At $m = M$, Equation (3.6) is the same as Equation (3.2).

Ensuring the posterior distribution is consistent

From Equations (3.5) and (3.6), the 'effective' posterior distribution for the m^{th} level in the cascade is

$$f^{(m)}_{X|Y}(x^{(m)}, y) \propto \exp\Bigg\{ -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{2^{2m}} \Big\{ \beta_1^{(m)} \sum_{k,l \in \delta^1_{ij}} I_{[x^{(m)}_{ij} \neq x^{(m)}_{kl}]} + \beta_2^{(m)} \sum_{k,l \in \delta^2_{ij}} I_{[x^{(m)}_{ij} \neq x^{(m)}_{kl}]} \Big\} - (2\sigma^2)^{-1} \sum_{i=1}^{n} \big(y_i - h_i(x^{(m)})\big)^2 \Bigg\}, \tag{3.7}$$

for $x^{(m)} \in \Omega^{(m)}$ and $m = 0, \ldots, M$. At $m = M$, Equation (3.7) is equivalent to Equation (3.3) and it is consistent with Equation (2.6).

3.3.3 The 2D discrete subpixel algorithm

The subpixel optimisation problem requires each pixel at $L^{(0)}$ to be subdivided into a $2^m \times 2^m$ array of subpixels in a single step. Our approach is to simplify this problem by breaking it down into a sequence of related but simpler optimisation problems. The subpixel cascade algorithm splits each pixel in the image $X^{(m)}$ at the m^{th} level into 2×2 blocks of subpixels that partition the image $X^{(m+1)}$ at the $m + 1^{\text{th}}$ level, for $m = 0, \ldots, M - 1$. At each level of the cascade, we search for the mode of the posterior distribution of $X^{(m)}$.

Algorithm 3.1. DISCRETE 2D SUBPIXEL.

1. *Set $m = 0$. Use the CMC of the record as the initial reconstruction, at level $L^{(m)}$.*
2. *Find the mode of the posterior distribution, in Equation* (3.7)*, using simulated annealing, including annealing at a temperature of zero to convergence (i.e. a strictly downhill optimisation).*
3. *Split each coarse pixel $X^{(m)}_{ij}$ into a block of 2×2 subpixels, where each of the four subpixels has the same initial colour as the original pixel.*
4. *Increment m by one.*
5. *Goto Step 2, until $m = M$.* □

To search over the posterior distribution, we use the optimisation algorithms outlined in the previous chapter (see §2.3.8 on page 33). That is, we apply simulated annealing for a finite number of sweeps, then use the image with the lowest energy to initialise the ICM algorithm, which is run to convergence. We use the Gibbs sampler to update the subpixels during each sweep of the image because the marginal distribution of a subpixel $X^{(m)}_{ij}$ takes only two values. In fact, it is computationally feasible to consider updating several pixels simultaneously. For example, to simultaneously update a 2×2 block of subpixels means sampling from a marginal distribution which takes 16 values (see §3.3.5 below).

The final image at each stage is used as the starting image for the next level of the cascade, by splitting each subpixel into a 2×2 block and setting each of those four pixels to be the same colour as the corresponding single pixel at the previous coarser level. These two images have different levels of resolution, one is of size $2^m \times 2^m$ and the other $2^{m+1} \times 2^{m+1}$. However, the value of $\phi(x)$ is the same for the last reconstruction at level $L^{(m)}$ and the first reconstruction at level $L^{(m+1)}$, prior to any processing at level $m+1$. (Note that the normalising constants of the p.d.f. for levels $L^{(m)}$ and $L^{(m+1)}$ are different.)

At lower-resolution levels in the cascade, there are fewer pixels so reconstructions are fast. The algorithm slows down at the finer resolutions, as there are more pixels to process. However, even with a very coarse image, say 32×32, the resolution improves very quickly, so that after level five the resolution has reached 512×512. Typically, less than six levels of reconstruction are needed. The process is much slower for grey-level images or if blurring has to be included.

3.3.4 Adjusting the temperature schedule while cascading

To ensure the algorithm does not wander from its initial starting point at the beginning of each level, its starting temperature is reduced from one level to the next. A typical starting temperature value is 20% of the starting temperature at the previous level but in practice this value must be checked by trial and error (see §3.4).

3.3.5 Single verses multiple site updating

During Step 2 of Algorithm 3.1, we can consider updating the Markov chain by changing one subpixel at a time or by changing several subpixels simultaneously because the marginal distribution takes only a few values.

At the m^{th} level, if single site updating is used then the colouring of each subpixel $X_{ij}^{(m)}$ is updated to minimise the current energy holding all other subpixels in the image fixed. A typical difficulty with single site updating samplers is that they mix, and hence converge, slowly in the presence of multi-modality. This might occur when the data are relatively un-informative compared to the spatially structured prior. In image analysis terms, this can occur when the algorithm relies on the prior distribution to reveal the fine detail in the image. In §3.4, an example is given where the record is on a coarse lattice which deliberately obscures some of the detail in the known, true image. This suggests that during the early levels of the cascade, the drop in the energy of the image comes mainly from improvements in the likelihood; in the final levels of the cascade, we might expect to rely more and more on the prior to reduce the energy.

To reduce the possibility of the algorithm getting trapped in local minima and to speed the propagation of information through the image, we also consider multiple site updating samplers. This may speed up the rate of convergence. We expect it to improve the quality of the reconstruction because it offers greater prospects of getting out of local minima by changing an entire block of subpixels in a single update. It is usually not practical to use the Gibbs sampler for simultaneous updates of small groups of conditionally dependent

components unless the state space of the marginal distribution is very small and discrete, but such is the case for the binary image in §3.4.

The naive solution treats each block of pixels $X_i^{(m)}$ as a single unit and updates all subpixels in this block simultaneously, for $m = 0, \ldots, M$. However, simultaneous updates for a *whole* block are feasible for levels $L^{(0)}$ and $L^{(1)}$ only because, at higher levels, the amount of computation required becomes prohibitive very quickly. With 2^m pixels in each block at the m^{th} level reconstruction, this means 2^{2^m} possible colourings; instead of 2^m searches over the two possible colourings with sequential updating. Such computational complexity precludes this naive attempt at multiple updating.

As a compromise, we simultaneously update subpixels $X_{ij}^{(m)}$ in disjoint 2×2 blocks. There are 16 colourings to consider with each update. This updating mechanism is not too expensive computationally, even for the Gibbs sampler. The benefit of this variation to the subpixel algorithm is that we expect to produce better results than single pixel updating.

Remark. We also considered simultaneous 3×3 updates. Using the Gibbs sampler, the amount of computation is substantially greater but still feasible, at least for small images. More generally, the Hastings-Metropolis algorithm could be used.

An example of the benefits of multiple site updating is shown in §3.4. Multiple updating of pixels is also used in Chapters 4 and 5, when groups of two or more pixels are updated simultaneously.

3.4 An application

3.4.1 The observed data and true image

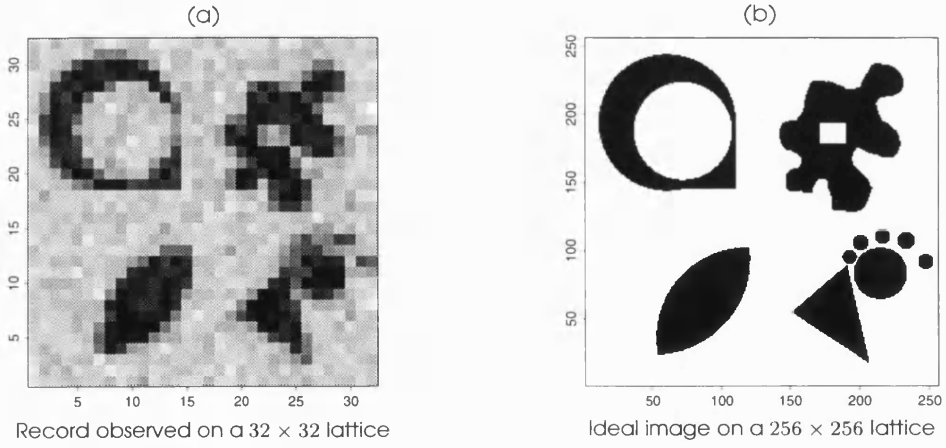

Figure 3-3: Discrete Subpixel Example — 1/3. Plot (a) shows the observed data, which is recorded on a 32×32 lattice. Suppose we decide to reconstruct the true image on a 256×256 lattice. Plot (b) shows what the reconstruction would be like on a 256×256 lattice, if we had perfect information.

The subpixel model is illustrated with a simulated example. Suppose we observe some continuous, binary image and the record suffers from additive Gaussian noise. In this example, the foreground colour, background colour and noise are know, $b = 1$, $w = 0$ and $\sigma^2 = 0.001$. In practice, they would have to be estimated from the data or from prior, expert knowledge. The record is observed on a 32×32 lattice and is shown as a grey-level image in Figure 3-3 (a).

Although the true image is assumed to be continuous it has to be discretised on some scale. For this example, the record is derived from an image that was digitised on a 1024×1024 lattice. This discretised, true image is sufficiently detailed to treat it as being continuous because each record element is derived from a mixture of 32×32 pixels in the original 1024×1024 image. Changing just one of the 32×32 pixels in the true image results in a change of less than a 0.1% in that record element. Thus the discretised nature of the *true* image has no influence.

We now want to reconstruct the true image to a greater detail than the observed 32×32 record. Suppose we decide to reconstruct the true image on a 256×256 lattice. Plot (b) shows what the reconstruction would be like, if we had perfect information. It is free from distortion because it has been derived directly from the true, 1024×1024 binary image, without any noise. Each pixel in Plot (b) is formed by thresholding the corresponding 8×8 block of pixels in the original 1024×1024 image.

This simulated example contains several features that are intended to stretch the limits of Algorithm 3.1. The fingers of the 'paw print' in the bottom right are very close together. It is not reasonable to expect the algorithm to differentiate them but we include them to see how well the algorithm can cope. Likewise for the thin, narrow, vertical strip of black in the top left object.

3.4.2 Using multiple site updating

We now apply Algorithm 3.1. The values of the smoothing parameters for the prior are $\beta_1 = 0.1$ and $\beta_2 = \beta_1/\sqrt{2}$, following Silverman and Jennison's recommendation for choosing the relationship between the first and second order smoothing parameters. In Figure 3-4, we use 2×2 multiple pixel updating. During each level, we apply 100 sweeps of the simulated annealing algorithm, then apply ICM to the simulated annealing reconstruction with the lowest temperature to convergence. A geometric temperature schedule with starting and stopping temperatures of 5 and 0.1 are used. The starting temperature is reduced by 80% at the start of levels 1, 2, and 3, to ensure that the algorithm doesn't deviate too far from its starting point. This value is chosen by trial and error, until the energy does not rise suddenly at the start of a new level.

We start on a 32×32 lattice, $L^{(0)}$, and finish on a 256×256 lattice, $L^{(3)}$. The final reconstruction for each level is shown in Figure 3-4 (a)–(d), respectively. The energy of the final reconstruction for each level have been standardised, so comparisons are valid. (This is because the smoothing parameters $\beta_1^{(m)}$ and $\beta_2^{(m)}$ in Equation (3.7) vary by level.) The steady drop in energy shows the advantage of subpixel reconstruction over reconstructions

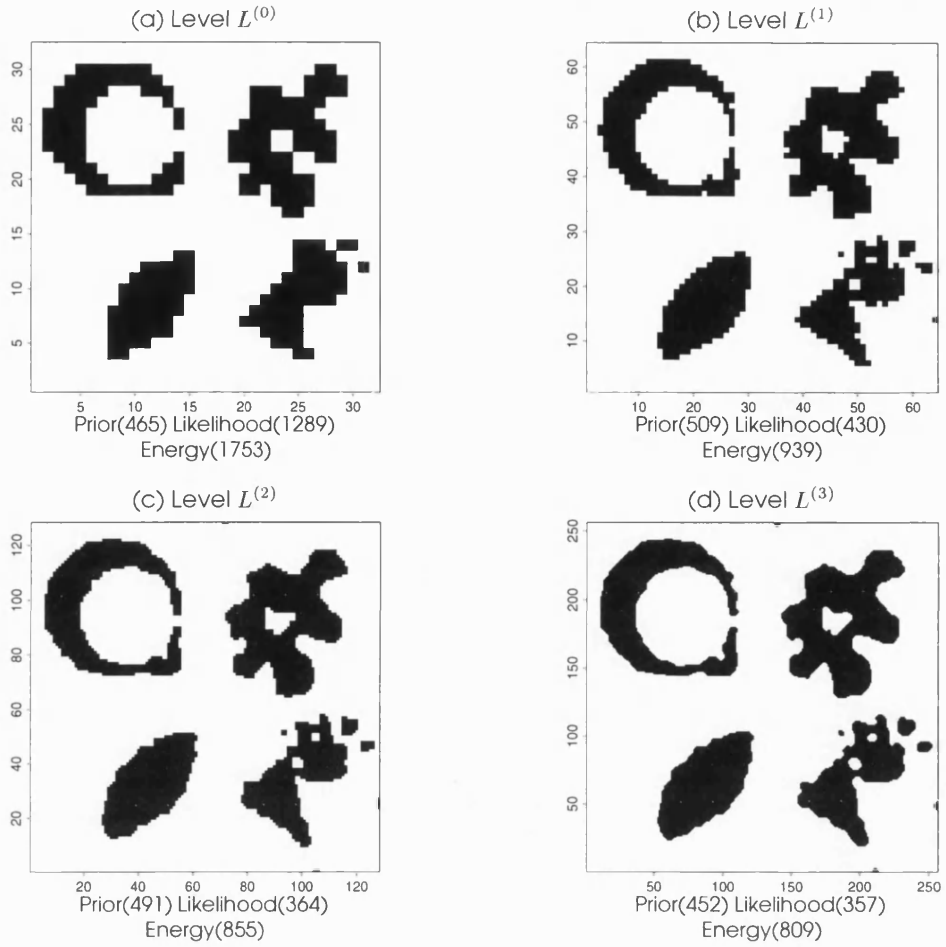

Figure 3-4: Discrete Subpixel Example: Multiple Updates — 2/3. Using multiple updates, simulated annealing and ICM, Plots (a)-(d) show the final reconstruction from each of the levels $L^{(0)}$ to $L^{(3)}$. The number of pixels increases fourfold between successive levels. The energy for each image is also shown. Plot (a) is the lowest energy reconstruction on the same lattice as the record. For finer and finer lattices, the energy of the final reconstruction decreases but the improvements are less pronounced. Another benefit of subpixel reconstruction is that level $L^{(3)}$ (Plot (d)) displays finer detail than level $L^{(1)}$ (Plot (a)), relative to Figure 3-3 (b).

at the original, 32×32 resolution. The decrease in energy is less pronounced at finer resolutions suggesting that there is a limit to the benefit of the algorithm. Each new level means a fourfold increase in the number of pixels and in the amount of CPU time required. This reconstruction took approximately 40 minutes of CPU time on a Sparc server 1000. Faster reconstructions are possible by reducing the number of simulated annealing sweeps. Runs with about fifty sweeps still produce good reconstructions.

3.4.3 Using ICM updating only

If we rerun the algorithm but ignore simulated annealing, we do not expect to do as well. Figure 3-5 (a) shows the final 256×256 reconstruction from ICM alone. All other parameters remain unchanged from those in the previous subsection including 2×2 multiple site

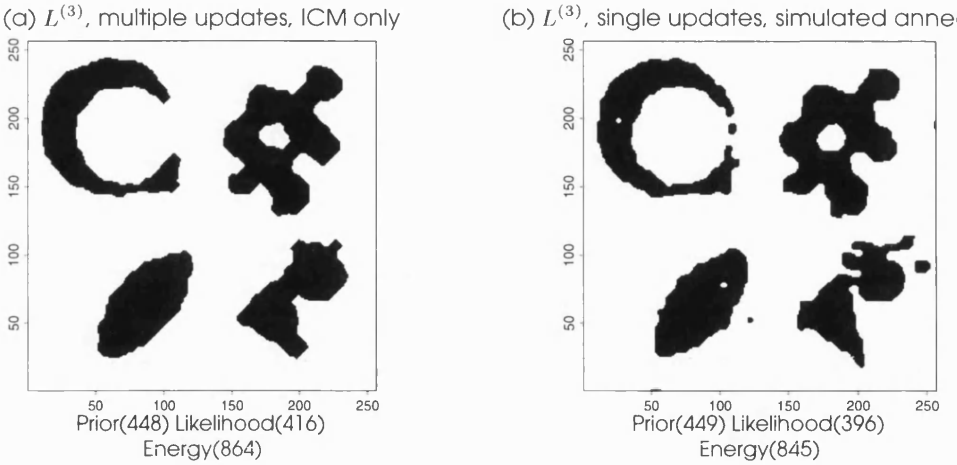

Figure 3-5: Discrete Subpixel Example: ICM Only and Single Updates — 3/3. The two images in this figure are similar to Figure 3-4 (d), except that Plot (a) uses multiple site updating but with ICM only (no simulated annealing) and Plot (b) uses single site updating (but with simulated annealing). Neither reconstruction does as well as Figure 3-4 (d). Plot (a) in particular loses much of the detail.

updating. This run of the algorithm is much faster than the previous run, where simulated annealing is also used. The average run using ICM only requires about four minutes CPU time. The cost is a poorer reconstruction. The final energy in Figure 3-5 (a) is only about 7% higher than in Figure 3-4 (d) but a visual inspection suggests that much of the finer detail in the ideal image (Figure 3-3 (b)) has been lost, relative to Figure 3-4 (d). This also suggests that relying on a single statistic to summarise an image, the energy of an image in this case, can be misleading.

3.4.4 Using single site updating only

It is worth considering the benefits of 2×2, multiple site updating. Figure 3-5 (b) shows the reconstruction when single site updating is used. All other parameters remain unchanged from those in §3.4.2, including 100 simulated annealing sweeps per level. Figure 3-5 (b) should be compared with Figure 3-4 (d) and Figure 3-5 (a). The paw print in the lower right-hand corner is reproduced more accurately using 2×2 multiple updates, such as in Figure 3-4 (d). This is because the individual paw prints are only a pixel or less in size at the 32×32 scale, as can be seen by comparing Figure 3-3 (a) and Figure 3-3 (b). Here is an example of where the data are relatively un-informative compared to the prior, as discussed in §3.3.5. In addition, the energy and the number of misclassified pixels (not shown) are lower in the multiple-pixel update reconstruction.

3.4.5 Comparisons between various runs

Table 3.1 shows some energy values for the three runs described above: simulated annealing and ICM combined with multiple site updating, simulated annealing and ICM

	(1) Multiple updates			(2) Single updates			(3) ICM only		
Level	Prior	Lklhd	Pstr	Prior	Lklhd	Pstr	Prior	Lklhd	Pstr
$L^{(0)}$	465	1289	1753	486	1269	1754	469	1253	1722
$L^{(1)}$	509	430	939	550	447	997	515	472	987
$L^{(2)}$	491	364	855	500	380	881	494	421	915
$L^{(3)}$	452	357	808	467	378	845	448	416	864

Table 3.1: ENERGIES BY LEVEL FOR VARIOUS RUNS. The prior, likelihood and posterior energies of the final reconstruction are shown for each of the four levels of the cascade. Typical results for three different runs are shown. The 'multiple updates' run uses simulated annealing and multiple site updating. The 'single updates' run uses both simulated annealing but with single site updating. The 'ICM only' run uses ICM only but with multiple site updating. As expected, run (1) produces a lower posterior energy than run (2) and run (2) does better than run (3).

combined with single site updating and ICM only combined with multiple site updating. For each run, the prior, likelihood and posterior energies are shown for the final reconstruction at each level.

Surprisingly, at the end of level $L^{(0)}$, ICM on its own has found a lower energy reconstruction than the other two runs. Even at level $L^{(1)}$, ICM with multiple updates has a slightly lower energy than single site updating with simulated annealing, followed by ICM to convergence. The final reconstructions have energy values that are as expected. Using simulated annealing and multiple site updating produces lower energy reconstructions than when single site updating is used. ICM does worst, even with multiple site updating.

For each of the three runs, the major drop in energy between the first two levels comes from a decrease in the likelihood energy. Between the last two levels, the decrease in energy comes from the decrease in the prior. These results are broadly in agreement with what we expected from §3.3.5.

3.4.6 Final comments about this example

This simple example tentatively suggests that this intuitive model could work well in practice. The principal disadvantage with the model and algorithm is that the amount of computation increases rapidly, especially if blurring is involved. One way of overcoming this computational burden might be to move from a discrete model to a continuous model. After all, the true image is believed to be continuous. This idea is explored in the next chapter.

3.5 The superpixel cascade algorithm

3.5.1 Background and notation

The motivation for the subpixel model comes from Jubb and Jennison's cascade algorithm, which we refer to as the 'superpixel' algorithm, to distinguish it from the subpixel model discussed in §3.2. It is not our intention to describe the superpixel cascade algorithm in

detail here; full details are available in Jubb and Jennison (1991). However, a brief outline, contrasting the two models, shows that the subpixel model described in this chapter is a distinct model in its own right.

The superpixel cascade algorithm is *not* a subpixel method. It was originally used as a simple and efficient multiple-site adaptation of ICM that is intended to deal with noisy $2^m \times 2^m$ blocks of pixels. Unlike the subpixel algorithm, this algorithm averages over blocks of pixels to form coarser and coarser reconstructions.

We generally use the same notation as in the previous section except that the superpixel cascade initially aggregates pixels rather than splitting them. So $m = 0$ is the original level at which the data are recorded. Let $Y^{(0)} \stackrel{\text{def}}{=} Y = (y_1, \ldots, y_n)$ and $X^{(0)} \stackrel{\text{def}}{=} X = (x_1, \ldots, x_n)$ be the record and the reconstruction on the original scale. At the m^{th} step, which we refer to as level $L^{(m)}$, $X^{(m)}$ is the reconstruction and $Y^{(m)}$ the corresponding record.

3.5.2 The superpixel cascade algorithm

During the first stage of the algorithm, m increases and the reconstruction becomes coarser. Both $X^{(m)}$ and $Y^{(m)}$ are formed by averaging the image X and the original record Y over the disjoint blocks of $2^m \times 2^m$ pixels. For simplicity and convenience, the authors use 2×2 blocking. At the coarsest level, we set $m = M$. If $n = 2^M \times 2^M$, there is only one pixel at $L^{(M)}$. Otherwise some adjustment may be needed for images with dimensions that are not a power of two.

Averaging over record values reduces the signal-to-noise ratio. The variance at level m is reduced by a factor of four relative to the next level, $\sigma^2_{(m)} = \sigma^2_{(m+1)}/4$, because we average over groups of four pixels to get from level $L^{(m)}$ to the level $L^{(m+1)}$. The variance at the original level $\sigma^2_{(0)}$ is estimated from the data.

The second stage is one of optimisation. We look at the sequence of records and reconstructions from the previous stage in reverse order. For each level $L^{(m)}$, where $m = M, M-1, \ldots, 0$, we seek the mode of the posterior distribution at that level. The posterior distribution for level $L^{(m)}$ is

$$f^{(m)}_{X|Y}(x,y) = \exp\Big\{ -\frac{1}{2}\beta \sum_{i \in X^{(m)}} \sum_{j \in \delta^1_i} I_{[x^{(m)}_i \neq x^{(m)}_j]} - (2\sigma^2_{(m)})^{-1} \sum_{i \in X^{(m)}} (y^{(m)}_i - x^{(m)}_i)^2 \Big\} \quad (3.8)$$

where I is the usual indicator function from Equation (2.15) and $\sigma^2_{(m)}$ is the variance of the noise at level m. Jubb and Jennison (1991) use the same MRF model at all grid levels. This means that the strength of association between pixels, denoted by β, remains unchanged at each level of the cascade.

The answer, from processing an image at level $L^{(m)}$ initialises the algorithm at level $L^{(m-1)}$, the next finer level of resolution. To get from $X^{(m)}$ to $X^{(m-1)}$, each coarse pixel is split into a block of 2×2 subpixels, where each of the four subpixels has the same initial colour as the original pixel.

Algorithm 3.2. FULL PIXEL CASCADE.

Stage 1: Aggregation and averaging

1. *Set* $m = 0$.
2. *Use the CMC of the record as the initial reconstruction.*
3. *Average non-overlapping* 2×2 *blocks of pixels to form an image with fewer pixels.*
4. *Increment* m *and repeat the previous step, until* $m = M$, *where* M *is the predetermined coarsest level of resolution.*

   Remark. This results in progressively coarser resolution images, each with successively higher signal-to-noise ratios. This process eventually stops because a one-pixel image is the limit. A sequence of fine-to-coarse images $X^{(0)}, X^{(1)}, \ldots, X^{(M)}$ has then been generated.

Stage 2: Cascading

1. *Initialise this stage of the algorithm to the final coarse reconstruction at the previous stage,* $X^{(M)}$.
2. *Apply ICM until convergence to find the mode of the distribution in Equation* (3.8).
3. *Split each pixel in* $X^{(m)}$ *into* 2×2 *blocks where all four subpixels are initially the same colour as the corresponding single pixel at level* $L^{(m)}$.

   Remark. The signal-to-noise ratio increases by a factor of four between successive levels in the cascade.
4. *Decrease* m *by one.*
5. *Goto Step 2 of this stage, until* $m = 0$.

   Remark. The reconstruction $X^{(0)}$ is back to the resolution of the original image data. □

3.5.3 Alternatives to the superpixel algorithm

Hurn (1992) extends the Jubb and Jennison (1991) algorithm to ensure consistent use of the prior model between different levels of the cascade and she also extends the underlying model to include the effects of blurring.

Other methods used instead of the cascade algorithm include: the re-normalised group approach (Gidas 1989), the Swendsen-Wang algorithm (Swendsen and Wang 1987), multigrid techniques and pyramid methods. See Hurn (1992) for a comparison of these algorithms.

3.5.4 Contrasting the subpixel and superpixel algorithms

The are some similarities between the subpixel cascade algorithm, in §3.3, and the Jubb and Jennison 'superpixel' algorithm. There are also many differences.

- The similarities between the two algorithms are: both algorithms proceed in stages, use the record as the initial starting point and split pixels into 2×2 blocks between each stage. The final image at each stage is used as the starting image for the next level of the cascade.

- In Jubb and Jennison, the energy is minimised at each level of the cascade using ICM, although the authors could also have used simulated annealing. At higher levels, there are fewer pixels and so less work is required. This algorithm is computationally cheap because the finest level of resolution is that at which the original data were recorded.

  The subpixel model uses simulated annealing and ICM so the algorithm takes longer to run but it searches for a global solution. The number of subpixels increases by a factor of four between levels, starting with the same resolution as the record. However, there are usually only three or four levels in the cascade. The user has the option of initially using only ICM to get an approximate but faster answer.

- Unlike Jubb and Jennison, there is no aggregation of pixels in the subpixel algorithm so the signal-to-noise ratio remains unchanged during the reconstructions. While this may appear to be a disadvantage, in practice it is a prerequisite to have a high SNR to be able to produce sensible subpixel reconstructions.

- The MRF prior in the underlying subpixel model changes from one level of the cascade to the next, in order to ensure consistency between the energies of the sequences of reconstructed images. In this respect, it reflects Hurn's adaptation of Jubb and Jennison's algorithm.

- In the subpixel algorithm, the reconstructions start on the same level of coarseness as the record and move from a coarse to fine resolution. Once the finest level of resolution has been reached the algorithm stops. This is unlike the Jubb and Jennison algorithm whose resolution goes from fine to coarse and then back to fine again, although they only process from coarse to fine.

3.6 Summary of chapter

The purpose of this chapter is to introduce a subpixel model that operates on a discrete lattice. An algorithm to implement the model is also defined. To achieve this:

- The idea behind the discrete 2D subpixel model is outlined.

- The prior, likelihood and posterior distributions for the model are defined for the finest level of resolution.

- It is possible to define a model for each level in the cascade. This leads to the constraint that the models must be consistent between the posterior distributions at different levels in the cascade. This is achieved by making adjustments to both the prior and the likelihood distributions, to make them dependent on the levels of the cascade.

- An algorithm to implement the model is specified. No assumptions are made about the shape of the objects in the image. The algorithm does require iteration. However, it does not rely on interpolation to get subpixel answers, as other authors have assumed.

  Multiple verses single site updating is discussed along with temperature adjustments to prevent the algorithm from wandering too far from its starting point.

- A simulated example is analysed, comparing single and multiple site updating. The example also compares simulated annealing and ICM results.

- Jubb and Jennison's full pixel algorithm is briefly outlined and contrasts are made to establish the subpixel model as a separate model. References to alternative methods are given.

Chapter 4

Continuous 2D subpixel reconstruction

Everything you've learned in school as
"obvious" becomes less and less obvious
as you begin to study the universe.
For example, there are no solids in the universe.
There are no absolute continuums.
There are no surfaces.
There are no straight lines.
Richard Buckminster Fuller

It's a small world,
but I wouldn't want to paint it.
Steven Wright

4.1 Introduction

4.1.1 Background to the algorithm

In the previous chapter, we proceeded in steps from a coarse to a fine lattice but each step required a substantial amount of computation. Also, each pixel in the discrete subpixel model may be split into many subpixels. In theory, each subpixel could be coloured differently to its neighbours, leading to several regions of colour within a single pixel. In practice, the prior prevents this undesirable possibility from happening. These facts suggest that it might be better to proceed directly to a continuous reconstruction, where each pixel is split into at most two regions separated by a straight line segment across the pixel and line segments link together across the image.

Example 4.1. BENEFITS OF CONTINUOUS SUBPIXEL RECONSTRUCTIONS.

For example, Figure 4-1 (a) and (b) re-show the continuous binary image and the corresponding observed image, from Figure 1-2. Figure 4-1 (c) shows a full pixel reconstruction where each pixel is either black or white. Figure 4-1 (d) shows the corresponding reconstruction obtained using the algorithm we describe in this chapter. Apart from a few errors at the edges of the image, the object boundaries are recovered to a high degree of accuracy. Ideally, the properties or position of the pixel grid should have a minimum effect

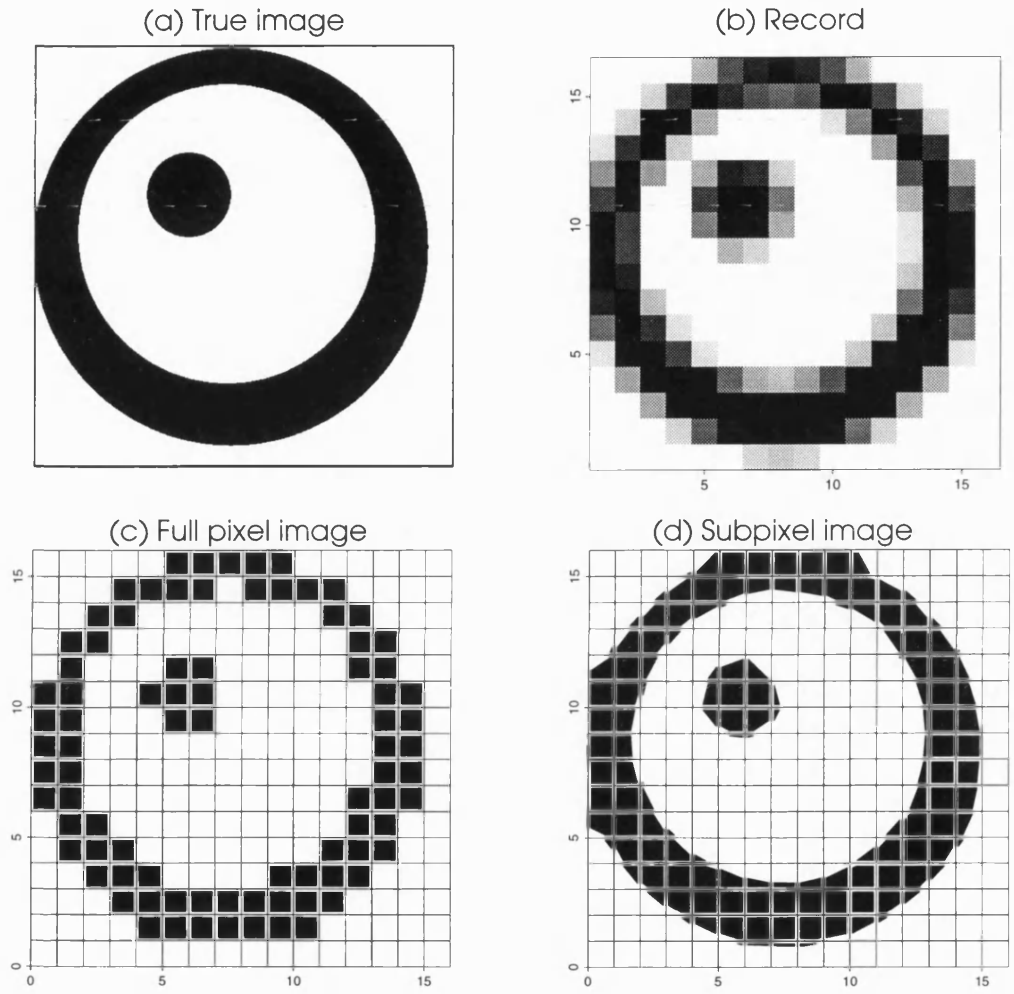

Figure 4-1: BENEFITS OF CONTINUOUS SUBPIXEL RECONSTRUCTIONS. A continuous binary image, the observed record shown as a 16×16 grey scale pixel image, a full pixel reconstruction and a subpixel reconstruction of the continuous image are shown in Figures (a)—(d) respectively. The true image shows two objects, one lying completely inside the other. Both objects are circular rather than straight-lined and the smaller object is only a few pixels in size. Figure (c) represents a typical full pixel reconstruction. In contrast, Figure (d) shows the benefits of a subpixel reconstruction, edges are more clearly defined. This facilitates accurate edge-length and surface-area estimation.

on the reconstruction of the true image. In Figures 4-1 (c) and (d), an artificial, pixel-size grid has been imposed to highlight individual pixels.

The model and algorithm used to generate the reconstruction shown in Figure 4-1 (d) are explained in §4.2 on page 67 and §4.3 on page 72, respectively. □

4.1.2 An overview

Just as with the discrete model in Chapter 3, we adopt a Bayesian approach, incorporating prior information about the true image in a stochastic model. We attach higher probability to images with shorter total edge length.

The overall objective is *to improve the quality of the restoration at a boundary, between two regions of different colour, by allowing each pixel to contain two regions of colour separated by a straight line.*

In reconstructions, pixels may be of a single colour or split between two colours. Unlike the previous chapter however, only two regions of colour are allowed and they are separated by a straight line. For this reason, we refer to this model as the *continuous 2D subpixel model.* In order to preserve smoothness, the edge that defines the boundary across a pixel is assumed to meet another edge in a neighbouring pixel or else it meets the image boundary. An example of such a reconstruction is given in Figure 4-1 (d).

In a similar fashion to the previous chapter, MCMC algorithms are then used in searching for the mode of the posterior distribution, which we take as our image estimator. The algorithm accepts as input an initial estimate where edges lie on pixel boundaries and, after a process of iterative refinement, outputs a more accurate estimate.

There are many ways in which a line segment can be placed across a pixel and there is spatial correlation between pixels. To give a precise location for the boundary, a complete mathematical model of the object's shape is not required but the space to be searched has many dimensions. This leads to a huge optimisation problem. In order to produce a tractable solution, only local information is used at any stage in the iterative calculations. This makes it easier to update the current image reconstruction. In addition, we break down the formidable problem of finding the mode of the posterior image distribution into stages and use the solution at the end of each stage as the starting point for the next. In subsequent stages, the reconstruction focusses on finer details but also avoids straying too far from its starting point.

Example 4.2. CONSTRAINED AND UNCONSTRAINED EDGES.
Figure 4-2 shows two stages of the reconstruction algorithm. On the left, edges are constrained to lie midway along a pixel boundary and neighbouring edges are linked to each other or to the boundary. The right figure shows the final stage of a reconstruction. Vertices are no longer constrained to lie on the midpoint of a pixel edge. □

Although this chapter is concerned with a 2D model only, the model is designed bearing in mind that we want to extend it to three dimensions (3D). This 3D model is described in Chapter 5.

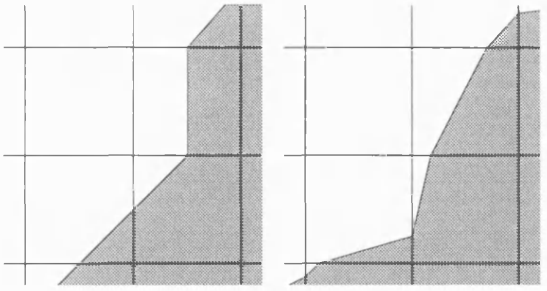

Figure 4-2: CONSTRAINED AND UNCONSTRAINED EDGES. The left figure shows an early stage of a reconstruction, where edges are constrained to lie midway along a pixel boundary. The right figure shows the final stage of a reconstruction. Each pixel may be split into two regions separated by a straight line.

4.2 The model

4.2.1 The prior distribution

For practical, computational and intuitive reasons, we allow at most a *single, straight-line edge* to divide any pixel between the two colours in our reconstructions and we incorporate this property into our prior model for the true, continuous scene. We also assume that edges are linked together. However, unlike Geman and Geman (1984), the location of edges is not constrained to lie only along the boundary between pixels. Edges are also located within pixels, splitting them into regions of different colours. Thus, we expect a fairly small number of regions of colour and smooth boundaries between them.

In this chapter, the prior distribution for the true scene X is on a class of binary images in which boundaries are continuous and piecewise linear.

Definition 4.1. We define the *image space* Ω to be the set of binary images satisfying the following conditions:

- Each 2D image X consists of an ordered, rectangular array of n pixels, $X = \{X_i : i = 1, \ldots, n\} \in \Omega$.
- Each pixel is either a single colour or divided into two regions of different colours separated by a single, straight-line edge.
- Each edge links with two edges in adjacent pixels, except at the image boundary.

Figure 4-1 (d) shows an example of such an image. Denote the colouring of pixel i by $h_i(x)$. We also use the notation $x(z)$ to denote the colour of image x at the point $z \in \Re^2$, in the true continuous image.

A MRF based on *edge length* defines a prior distribution for the scene X. Edge length is the length of the boundary between regions of different colours. The regions of different colours may lie in neighbouring pixels, within the same pixel or they may overlap pixels.

Model 4.1. PRIOR. *The prior energy for an image is based on the total edge length in that image, $L(X)$. The total edge length is the sum of line segments across pixels plus the sum of the edges lying on the boundary between neighbouring pixels of different colour.* □

So the prior probability for the scene X is

$$f_X(x) \propto \exp\{-U_X(x)\} \stackrel{\text{def}}{=} \exp\{-\beta L(x)\} = \exp\left\{-\beta \sum_{i=1}^{n} L_i(x)\right\}, \quad x \in \Omega, \tag{4.1}$$

from Equation (2.3). Lower values of the smoothing parameter β lead to a greater probability of objects in the image having long, usually jagged, edges.

Remark.

- In this chapter, we use a first order neighbourhood only. So the colouring for a pixel is conditionally independent of all other pixels, given the colouring of its *four* horizontally and vertically adjacent neighbours.
- This prior can be extended to three dimensions. In two dimensions, an image is a two dimensional array of pixels; the corresponding image in three dimensions is a three dimensional array of voxels. Instead of edge length across pixels, we can consider the surface area passing through voxels (see Chapter 5 on page 98).

What is the prior distribution a density with respect to?

Consider a distribution on the image space Ω defined by Equation (4.1). Let $i = 1, \ldots, N$ index all possible routes of edges that are allowed under Definition 4.1, each having probability $1/N$. For each possibility i, define a vector θ of the appropriate length with elements in the open interval $(0, 1)$ to locate the vertices of the edges along the pixel boundary. (This requires the direction in which the distance is measured to be recorded.) Conditional on a specified route i, let each element of θ have an independent uniform distribution on $(0, 1)$.

The distribution for the pair (i, θ) will not penalise edge length but it is useful for calculating likeliood ratios, such as appear in the acceptance probability of the Metropolis-Hastings algorithm. The density f in Equation (4.1) is with respect to the above measure on Ω. That is, we have a distribution on the same space as the (i, θ) pairs above but the likelihoood ratio of the distribution in Equation (4.1) to the distribution for the pair (i, θ) is f_X. In our particular case, the density of θ is 1 with respect to the space of possible values of θ for a particular route i because we know the distributions of the individual elements of θ. Our image model has density proportional to $\exp\{-\beta L(x)\}$ with respect to this distribution (measure) of θ. When running an MCMC method for this distribution, or for the posterior derived from it, the density is with respect to this underlying measure.

Non-degeneracy in the prior

Once the colouring of a pixel's four horizontal and vertical neighbours is known, the colouring of that pixel is also known. This is because the location of any edge across that pixel has to link up with its neighbours. To establish a Markov random field property for our model, without such degeneracy, consider a set of pixels in some neighbourhood of pixel i, δ_i. Define δ_i' to be the set of pixels which are horizontal and vertical neighbours

of the pixels in δ_i, excluding the pixels in the set δ_i itself,

$$\delta_i' \stackrel{\text{def}}{=} \{j \in \{1, \ldots, n\} : j \in \delta_{\delta_i}^1 \text{ and } j \notin \delta_i\},$$

where $\delta_\cdot^1$ defines a first order neighbourhood (see §2.2.3 on page 22). Then it can be shown, under Equation (2.2), that the set of pixels in a neighbourhood around pixel i is conditionally independent of the set pixels in X but not in δ_i or δ_i', given the set of pixels in δ_i',

$$\delta_i \perp\!\!\!\perp \{j \in \{1, \ldots, n\} : j \notin \delta_i \cup \delta_i'\} \mid \delta_i'.$$

We illustrate this point with the following example.

Example 4.3. NON-DEGENERACY IN THE PRIOR DISTRIBUTION.
Defining the prior for a single pixel gives a conditional distribution that takes just one

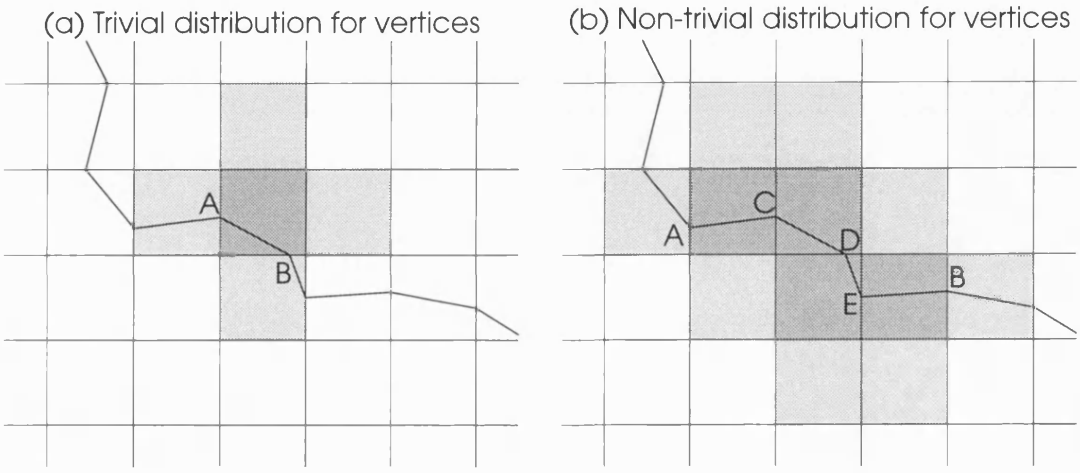

Figure 4-3: NON-DEGENERACY IN THE PRIOR DISTRIBUTION. The set of pixels of interest, δ_i, are coloured in dark grey and the set of neighbouring pixels, δ_i', is coloured in light grey. In Plot (a), the set δ_i consists of a single pixel. So the distribution of the vertices that it contains, A and B, is trivial, given the four neighbouring pixels. For the corresponding set of several pixels in Plot (b), the distribution of the vertices that it contains, A to E, is non-trivial because C, D and E move freely, even though A and B are still fixed.

value. This is because the ends of the line segment across the pixel are fixed once the four neighbours are known. Instead, the way to look at the distribution of the edges is to consider a set of pixels, δ_i, around some pixel i. The conditional distribution for δ_i given its neighbours depends on the location of the vertices of the two edges that lie on the boundary of the set. It also depends on what happens as the edges traverse across the set. Because there are many possible choices, the distribution is not trivial.

Figure 4-3 considers the distribution of vertices contained in the set of pixels δ_i where δ_i is the single, dark-grey pixel in Plot (a) and δ_i consists of the four, dark-grey pixels in Plot (b). The pixels in δ_i' are coloured in light grey. There are four such pixels in Plot (a)

and eight in Plot (b). So the light grey pixels δ_i' are the pixels which are horizontal and vertical neighbours of the set of dark grey pixels δ_i, excluding the pixels in the dark grey pixels themselves. (In this example, the pixel colours reflect the neighbourhood structure in the prior model; not the colouring in the image reconstruction.)

In Plot (a), for a set consisting of a single pixel, $\delta_i = \{i : \text{for one } i \in \{1,\ldots,n\}\}$, the distribution of the vertices that it contains, A and B, is completely determined once the set of neighbouring pixels is known. This is because A and B lie on the boundary of the set δ_i. If δ_i contains more than one pixel, shown in Plot (b), the distribution of the vertices that it contains, A to E, is non-trivial because C, D and E move freely. (The vertices A and B are remain fixed because they still lie on the boundary between δ_i and δ_i'.) □

4.2.2 The likelihood function

The observed data $Y = (y_1,\ldots,y_n)$ are recorded on a regular lattice of points $\{z_i \in \Re^2 : i = 1,2,\ldots,n\}$. In the absence of blurring, the expected value of the sensor's output at each Y_i is proportional to the average intensity within that pixel and is assumed to be proportional to the area covered by the object in that pixel. We denote the average value of the colour of pixel i by

$$h_i(x) = bp_i^b + wp_i^w, \tag{4.2}$$

where p_i^b and p_i^w are the proportions of pixel i covered by colours b and w respectively,

$$p_i^b = \iint_{z\in i} I_{[x(z)=b]}dz \quad \text{and} \quad p_i^w = \iint_{z\in i} I_{[x(z)=w]}dz.$$

(For convenience, we assume the area of a pixel is one.)

Due to imperfections in the recording sensor, the record Y may be degraded by blurring and additive noise. So an additive Gaussian model, with blurring, is assumed for the record.

Model 4.2. LIKELIHOOD. *The likelihood model is*

$$y_i = \iint x(z)g(z_i,z)dz + e_i \tag{4.3}$$

where $x(z)$ is the value of the true image at the point $z \in \Re^2$, $g(z_i,z)$ is a blurring function which decreases in value as the distance between z and the point z_i increases and e_i is independent, additive sensor noise. □

In practice, we use a discrete approximation to the blurring kernel in which $g(z_i,z)$ is assumed to be piecewise constant in z across pixels. So if blurring is present, the colour observed at pixel i is a combination of the colours in a neighbourhood around that pixel.

Equation (4.2) is then replaced by

$$h_i(x) = \sum_{j:j\in\delta_i} (bp_j^b + wp_j^w)K_{ij} \tag{4.4}$$

where δ_i is the set of pixels in a neighbourhood around pixel i. In effect, the colour in neighbouring pixels 'leaks' into pixel i. The amount of influence each neighbouring pixel has depends on its spatial distance from pixel i, which is stored in a *blurring matrix* K.

Remark.

- The size and shape of the blurring kernel determines the size of the matrix K and the value of the elements in the matrix. It is usually estimated from the data (see §4.4 on page 83).

- When blurring is present, pixels around the border of the image are only observed indirectly through their contributions to the records of neighbouring pixels. In this case, there are fewer elements in the record Y than there are pixels in the image X.

- Following on from the comments in §2.3.5 (on page 31), we assume that the blurring is shift invariant. Spatial stationarity is a standard assumption in image models. It removes the need for experimentation with proposal distributions tailored to individual variables, representing individual pixels.

The blurring coefficients K and the noise variance σ^2 are assumed to be known or can be estimated from the data. From Equation (4.4), the distribution of the the signal y given the true image x is then

$$f_{Y|X}(y,x) \propto \exp\Big\{-(2\sigma^2)^{-1}\sum_{i=1}^{n}\big(y_i - h_i(x)\big)^2\Big\}. \tag{4.5}$$

4.2.3 The posterior distribution

As usual, the posterior is the combination of the prior and likelihood, which in this model is

$$f_{X|Y}(x,y) \propto \exp\Big\{-\beta L(x) - (2\sigma^2)^{-1}\sum_{i=1}^{n}\big(y_i - h(x_i)\big)^2\Big\}. \tag{4.6}$$

We want to find the image x in the sample space Ω which maximises the posterior probability $f_{X|Y}(x,y)$ defined by Equation (4.6).

4.3 The algorithm

4.3.1 An overview

We use MAP estimation to find the image x in the sample space Ω which maximises the posterior probability $f_{X|Y}(x,y)$ defined by Equation (4.6). The sample space Ω, as defined in §4.2.1, is complex and it is difficult to find a good initial estimate in Ω directly. Instead, we proceed in stages, optimising first over simpler image spaces in order to obtain a good starting image in Ω for the final optimisation. At each stage we seek the maximum of

$$\phi(x) = \exp\{-\beta L(x) - (2\sigma^2)^{-1}\|y - h(x)\|^2\} \tag{4.7}$$

over images x in specified spaces, where $L(x)$ is the total edge length in image x and $h_i(x)$ the average value of x over pixel i. Our algorithm for seeking the MAP estimate is summarised below.

To aid the search, a cascade approach to optimisation is used, where each step in the cascade involves optimising over a different state space. The answer from each stage forms a starting point for the subsequent stage. There are four stages in all. Stages 1 to 3 are intended to produce a good starting image $x \in \Omega$, for the final stage. In Stages 1 and 2, we work with x in spaces different from Ω. Stage 1 has state space $\Omega_1 = \{x : x_i \in \{b, w\}\}$. Stage 2 has state space Ω_2 where, for $x \in \Omega_2$, each x_i takes one of the fourteen states shown in Figure 4-4. This introduces edges into the reconstruction but they are not necessarily linked. In Stage 3, Ω_3 is similar to Ω_2 but is constrained so that edges link together. Finally in Stage 4, the state space $\Omega_4 \equiv \Omega$ allows each vertex to be located anywhere along the pixel boundary on which it is initially located. The set of all possible such locations defines this state space Ω_4.

The relationship between the state spaces is $\Omega_1 \subset \Omega_2$, $\Omega_2 \supset \Omega_3$ and $\Omega_3 \subset \Omega_4 \equiv \Omega$. So the state space is increased in going from Stage 1 to Stage 2, reduced during Stage 3 and then increased again in Stage 4. State spaces Ω_1, Ω_2 and Ω_3 are easier to work with than Ω initially and the answer from each stage forms a starting point for the subsequent stage. This reduces the amount of work during each stage and ensures that each stage has a good starting point.

Remark.

- Within each stage, the algorithm relies on the Gibbs sampler and the Metropolis algorithm to simulate from the posterior and optimisation is performed using simulated annealing. Details are deferred until §4.3.2.

- Note there is something similar here to the previous chapter, the objective function is fixed and we maximise it over a series of related image spaces.

4.3.2 The 2D continuous subpixel algorithm

We now define the algorithm.

Algorithm 4.1. CONTINUOUS 2D SUBPIXEL.

Stage 1: Full pixel reconstruction *Define Ω_1 to be the set of images in which each pixel is of a single colour, b or w. From a convenient starting point, search for the image in Ω_1 that maximises $\phi(x)$, in Equation* (4.7) *using simulated annealing based on the Gibbs sampler (see §2.3.3 on page 29).*

Remark. The starting image is not critical for simulated annealing, if we use a sufficiently high temperature early on. So for convenience, the observed image is thresholded at a value midway between the foreground and background colours, which are estimated from the data.

Stage 2: Initial subpixel estimation *Define Ω_2 to be the space of images in which each pixel takes one of the fourteen states shown in Figure 4-4. Starting with the final reconstruction from Stage 1, search for the image in Ω_2 that maximises $\phi(x)$, again using simulated annealing based on the Gibbs sampler.*

Remark. This stage introduces edges into the reconstruction. For computational efficiency, an edge lying across a pixel is initially constrained to have vertices that lie midway along a pixel boundary. This results in the fourteen possible proposals shown in Figure 4-4. The edges in neighbouring pixels are not necessarily linked. This allows the algorithm greater freedom initially that is not allowed in the final stages when edges across pixels are piecewise continuous. An example of a 'fourteen-proposal' scene is given in Figure 4-5 (b).

Stage 3: Conversion to an image in Ω *Apply a deterministic algorithm (see §4.3.3 on page 77) to convert the end product of Stage 2 to an image in Ω with as little modification as possible.*

Stage 4: Final subpixel estimation *Starting from the reconstruction obtained in Stage 3, search for the image in Ω that maximises $\phi(x)$. This time a form of simulated annealing based on the Metropolis algorithm is used (see §4.3.4 on page 79).*

Remark.

- The boundary around each object in the image is now identified by a linked, sequence of straight lines, each of whose vertices initially lie midway along a pixel edge. The final stage allows the vertices of an edge to be located anywhere along the pixel edge on which it currently lies, except right on a corner.
- The search for the image $x \in \Omega$ is by simulated annealing with a Metropolis update, for moving the vertex at which two edge-segments meet along a pixel boundary. One advantage of this constraint is that the starting temperature can be set to a high value, in order to explore the state space Ω, because it is not possible for an edge to move more than a half a pixel from the starting point.

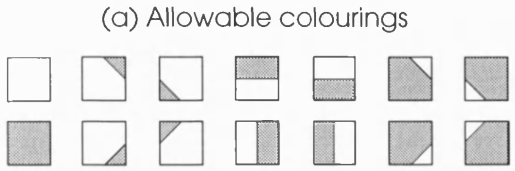

Figure 4-4: STAGE 2 PROPOSALS. The fourteen pixel colourings allowed in Stage 2 are shown. These possibilities arise from constraining every edge segment to terminate midway along a pixel boundary.

- An implicit multiple update occurs during this stage. When the edge vertex shared by two pixels is moved, both pixels are updated simultaneously. □

Example 4.4. RECONSTRUCTIONS FROM EACH STAGE OF ALGORITHM 4.2 — 1/2. Examples of the types of images produced in the four stages are shown in Figure 4-5.

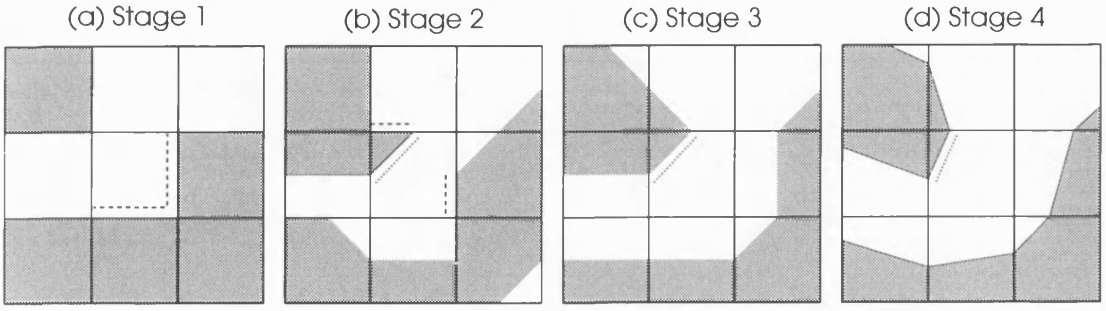

Figure 4-5: RECONSTRUCTIONS FROM EACH STAGE OF ALGORITHM 4.2 — 1/2. Each 3×3 image is an example from the state space used during the four stages of Algorithm 4.1. In each image the centre pixel's contribution to the edge length $L(x)$ is highlighted using dashed lines for edges lying on the pixel boundary and dotted lines for the edge inside the pixel. We are interested in reconstructions that lie in Ω, which arise in Stage 4.

The edge length penalty for the centre pixel in each 3×3 image is highlighted by broken lines. The dashed line shows the edge lying on the pixel boundary and the dotted line shows the edge lying inside the pixel. Stage 1 has edges lying on the pixel boundary only. This allows the algorithm to make large jumps over the space of possible images, by only allowing one colour within a pixel. Stage 2 allows straight lines across a pixel with the constraint that the vertices of the straight line are located midway along a pixel edge. This initial approximation substantially reduces the computational burden and is used to locate the boundary to within half a pixel of its final position. To encourage exploration of the state space, line segments are not necessarily linked together at this stage. Stage 3 ensures the line segments lying around the boundary of an object are piecewise continuous by linking edges together. Stage 4 relaxes the constraint that edges lie midway along a pixel edge. This leads to subpixel changes in the reconstructed image by allowing the vertices of each line segment to vary by up to half a pixel from their initial location. □

Example 4.5. RECONSTRUCTIONS FROM EACH STAGE OF ALGORITHM 4.2 — 2/2.
As a more substantial example, Figure 4-6 shows the four stages in the reconstruction of

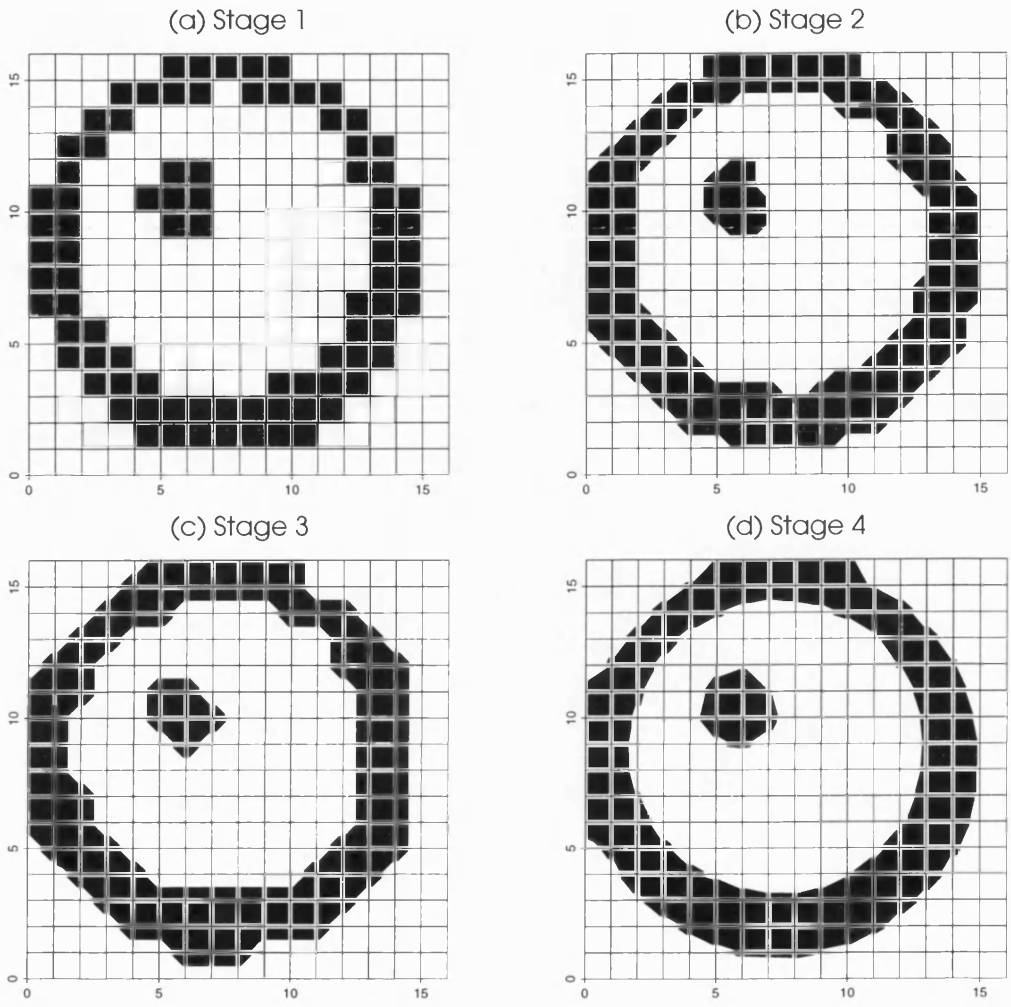

Figure 4-6: RECONSTRUCTIONS FROM EACH STAGE OF ALGORITHM 4.2 — 2/2. In Stage 1, each pixel takes one of two colours. In Stage 2, each pixel is permitted to take one of the fourteen states, shown in Figure 4-4, with no further restrictions. Only a few places require adjustment in Stage 3, in order to obtain the final reconstruction shown in Stage 4. Note that the final reconstruction has a tendency to 'cling' to the boundary as this reduces the total edge length within the image, so increasing $\phi(x)$. The benefit of a subpixel reconstruction is the difference between the Stage 1 and the Stage 4 reconstructions.

the image depicted in Figure 4-1. The record Y used here is the 16×16 array of the grey-level values in Figure 4-1 (b). Independent normal errors of mean zero and variance $\sigma^2 = 0.01$ have been imposed. The parameter in the prior image model, in Equation (4.1), is set at $\beta = 25$. The algorithm is initialised at the beginning of Stage 1 by thresholding the record Y, at the value midway between the two colours $(b+w)/2$. Simulated annealing with a geometric cooling schedule is used in each of Stages 1, 2 and 4.

For each of the reconstructions in Figure 4-6, a pixel grid has been superimposed and the region of black in each pixel has been drawn to lie *inside* the boundary of each pixel. This is purely a graphical device to help illustrate how the image is constructed. It does not affect the calculations.

A thresholding of the discretised image in Figure 4-1 is shown in Figure 4-6 (a). The reconstructions at the end of Stages 2, 3 and 4 are also shown. In Stage 2, edges which contravene the rules defining Ω are liable to increase $L(x)$ and decrease $\phi(x)$. Thus, it is not surprising that the Stage 2 image infringes these rules in only a few isolated instances on the edges of the ring and in several contiguous pixels on the right-hand side of the small black circle. These points are rectified in Stage 3 and further processing in Stage 4 yields a reconstruction which approximates the original image much better than the corresponding full-pixel reconstruction in Stage 1.

The most noticeable errors occur where boundaries between black and white regions meet the image edge. Here, distortions can arise as our prior model penalises edges running close and almost parallel to the image edge and favours more abrupt termination of such edges at an earlier point. A different prior may not have this characteristic feature. For instance, the prior could be based on the angle between edges, favouring straight line edges over jagged edges. □

Simulation for Stages 1 and 2

In Algorithm 4.1, an analytic approach to maximising $\phi(x)$ within each stage is not feasible because of the high dimensionality of the image spaces under consideration. Instead, we use the stochastic optimisation algorithm of simulated annealing at each stage (see §2.3.8 on page 34).

Simulated annealing based on the Gibbs sampler is used in Stages 1 and 2 of Algorithm 4.1 to seek the maximum of $\phi(x)$ over sample spaces Ω_1 and Ω_2 respectively (see §2.3.3 on page 29). In each iteration $t = 1, 2, \ldots$, the image is swept in a raster scan and pixels are updated in turn. Let x_{-i} denote the values of x at all pixels other than i, in the image x. If the current image is x just before pixel i is updated, the Gibbs sampler would generate a new image x' with probability proportional to $\phi(x')$ for all x' in the sample space satisfying $x'_{-i} = x_{-i}$. Thus, in Stage 1 pixel i can take either of the colours black or white and in Stage 2 pixel i can take any of the fourteen possible colourings shown in Figure 4-4.

In our example, we used a geometric temperature schedule falling from initial temperature 5 to final temperature 0.1 in 50 sweeps. Following Geman et al. 1987, the image attaining the maximum value of $\phi(x)$ during the 50 sweeps was noted and the algorithm was rerun at a temperature of zero from this starting point, until convergence at a local maximum of $\phi(x)$. This final step is equivalent to applying the strictly downhill search of iterated conditional modes (Besag 1986) from the starting point formed by simulated annealing.

4.3.3 Stage 3: The conversion algorithm

An image in Ω_2 can be fully specified by stating the colour present in each corner of each pixel. Of the eighteen possible combinations of two colours separated by a straight line, only fourteen are permitted. These are shown in Figure 4-4. The other possibilities are excluded because of the restriction to only one line segment crossing any one pixel. If an image in Ω_2 is also in the set Ω, the same colour must be associated with every point in the image where four pixel corners meet or where two pixel corners meet around the boundary of the image. It is straightforward to check that this is a sufficient condition for an image in Ω_2 to be in Ω and we base our conversion algorithm on this fact.

Algorithm 4.2. CONVERSION FROM Ω_2 TO Ω.

Initialisation *Use the final reconstruction from Stage 2 to assign a colour to each of the four corners of each pixel.*

Sweep 1 *Visit each pixel corner in turn and note the colours of the four adjacent pixel corners or two adjacent corners round the boundary, except at the corners.*

- *If all corners have the same colour then fix the corner values for that vertex.*
- *If one colour dominates by occupying three out of four corners then re-colour the minority corner to agree with the other three.*
- *If the colours are evenly split, one to one or two against two, calculate the average value of the records associated with all the pixels concerned and colour all corners with the colour that lies closest to the average record value.*

  Remark. Here we take the record associated with a pixel to be the record to whose mean the pixel makes the largest contribution.

Sweep 2 *Visit pixels in a raster scan taking rows from the top of the image to the bottom and moving from left to right within each row.*

- *If the four colours in the corners of a pixel correspond to a pixel colouring in Figure 4-4, assign this colouring to the pixel.*
- *If not, we must have one pair of diagonally opposite corners of one colour and the other pair of the other colour. To rectify this, switch the value in the pixels bottom right-hand corner and also the values in the corner of each neighbouring pixel which meet at this vertex. The pixel currently being visited now has a pattern of corner values in agreement with a colouring in Figure 4-4 and we assign it this colouring.* □

Example 4.6. THE CONVERSION ALGORITHM.
Figure 4-7 shows the conversion of the example in Figure 4-5 from Stage 2 to Stage 3 using Algorithm 4.2. The colour of each corner of each pixel, from Stage 2 of Figure 4-5, is used to initialise Figure 4-7 (a). During the first sweep, each corner of each pixel along with its

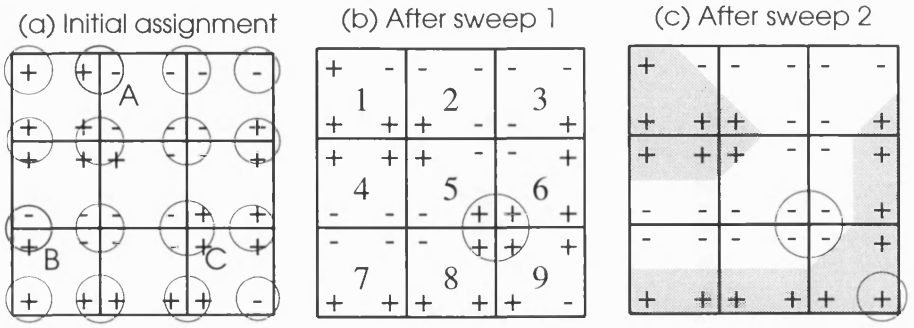

Figure 4-7: THE CONVERSION ALGORITHM. Figures (a)-(c) show the conversion of Figure 4-5 from Stage 2 to Stage 3 using Algorithm 4.2. In (a), the circles denote the grouping of corner values at pixel vertices. During Sweep 1, rules are applied to ensure that all corners meeting at a vertex have the same colour. During Sweep 2, the corners within each pixel are used to decide the colouring for that pixel. To avoid an anomaly in pixel 5, the colour of the South-East corner and its neighbours are flipped. This switch has a knock-on effect on pixel 9, so that the South-East corner of that pixel also changes.

neighbouring corners is processed and the circles in Figure 4-7 (a) highlight the corners involved in each step. For the circles labelled A, B and C, there is a tie between the two colours. This is resolved by considering the average of the records associated with those pixels.

During the second sweep of the image, the four corners within each pixel are considered and one of the fourteen proposals shown in Figure 4-4 (a) is inserted, based on the colouring of the four corners within that pixel. An ambiguity arises at the pixel labelled 5, in Figure 4-7 (b), because the colouring is not one of the possibilities shown in Figure 4-4. The South-East corner of pixel 5 is flipped and this has a knock-on effect on pixel 9, which now has a pattern of colours similar to that of pixel 5. After switching the South-East corner, we obtain the result shown in Figure 4-7 (c) which is the Stage 3 reconstruction of Figure 4-5. □

Avoiding Stage 3

The state space Ω_2 includes images where edges may lie along a pixel boundary. It is likely that edges will settle into a 'linked' pattern during the latter part of Stage 2 since this will tend to increase the posterior probability. The intention in these first two stages is to produce an initial approximation to the MAP subpixel estimate in an unconstrained search over the spaces Ω_1 and Ω_2.

There is no guarantee that the reconstruction at the end of Stage 2 will be in the set Ω, as edges of regions of a particular colour may lie along pixel boundaries rather than within pixels. Geman, Geman, Graffigne, and Dong (1990) suggest imposing an increasing penalty for undesirable or 'taboo' states to help ensure that edges across pixels are linked together. In effect, they added closed boundary constraints with increasing penalties for

each step in the algorithm. In our case the 'taboo' state would be a corner with *both* plus and minus around it. They show that, theoretically the constrained optimisation algorithm would converge to the optimal closed boundary configuration. But in practice, they admit that a 'linked' pattern is not always guaranteed from simulated annealing with a finite number of sweeps, even if a large penalty is attached to images that are not 'linked'. So to guarantee that edges are linked a separate stage for tidying-up may be needed.

To cater for this possibility, we use a convenient procedure that requires only one or two sweeps of the image, to amend any features which contravene the rules obeyed by images in Ω. After Stage 3, all edges lie within pixels and their vertices are at the midpoints of the pixel edges. From this point, the pixel-to-pixel route of each edge is fixed. Thus, the end product of Stage 3 determines a subset of Ω to be searched in Stage 4, when the vertices of each edge segment are allowed to move within their specified pixel edges.

Stage 3 could be unnecessary if we allowed up to two edges across a pixel. This would require 18 proposals to be considered. The four new proposals arise from pixels with both black and white located at diagonally opposite corners. This adjustment would guarantee that any Stage 2 reconstruction will lie in Ω, provided that Ω also allows pixels with up to two edges across it. However, this would lead to a more complex model. In particular, it implies that the analogous model in 3D could have four surfaces inside a voxel. This level of complexity is not desirable so this route is not explored further.

Any other conversion algorithm could be used as long as it does not upset the answer from Stage 2 too much.

4.3.4 Stage 4: Subpixel adjustments

At the end of Stage 3, the boundary around each object in the image is identified by a linked, sequence of straight lines, each of whose vertices lies midway along a pixel edge. So, the colouring of the image is specified up to the location of the vertices of the linked edge segments. In the final stage of our algorithm, we allow the vertices of an edge to be relocated anywhere along the pixel edge on which it lies initially. We do not allow vertices to move into neighbouring pixels. Thus, the image space is effectively reduced to a subset of Ω by the beginning of Stage 4. This reflects the work done during the earlier stages. One consequence of this is that the starting temperature for a simulated annealing search can be set to a high value in order to explore this subspace of Ω fully. Details of this application of simulated annealing are described in §2.3.3. As the Stage 4 image space is continuous, it is convenient to use simulated annealing based on the Metropolis-Hastings algorithm.

Suppose the vector $\theta = (\theta_1, \ldots, \theta_r)$ contains parameters, taking values in $(0, 1)$, which specify the location of the vertices. Let $x(\theta)$ denote the image defined by θ, let θ_{-i} denote the values of θ in all elements other than i and let θ' be the vector with i^{th} element θ'_i and $\theta'_{-i} = \theta_{-i}$.

The goal is to find the value which maximises $\phi\{x(\theta)\}$ over $\theta \in (0, 1)^r$. The Metropolis algorithm samples from the distribution of θ with density proportional to $\phi\{x(\theta)\}$.

Algorithm 4.3. SWEEPING AN IMAGE IN Ω.

- *Sweep the image, updating every vertex location once. In effect, the vector θ is swept and its elements θ_i are updated in turn, where $i = 1, 2, \ldots, r$.*

  Remark. Typically the number of updates required to sweep an image is far less than the number of pixels in the image, $r \ll n$, as only a few pixels contain two colours. This reduces the amount of computation required.

- *In updating θ_i, a proposal value θ_i' is drawn from the uniform density on $(0, 1)$. Replace θ_i by θ_i' with probability*

$$\alpha(\theta, \theta') = \min\Big[1, \phi\{x(\theta')\}/\phi\{x(\theta)\}\Big] \tag{4.8}$$

  otherwise the value θ_i is retained.

- *Increment i by one and goto the previous step to update the next element θ_{i+1}, until $i = r$.* □

The effect of changing a single θ_i is illustrated in Figure 4-8. The colouring of *two* pixels is *simultaneously* affected as the two edge-segments meeting at the vertex in question are repositioned. The only modification with the simulated annealing algorithm is the replacement of Equation (4.8) by

$$\min\Big[1, \Big\{\phi\{x(\theta')\}/\phi\{x(\theta)\}\Big\}^{1/T(t)}\Big]. \tag{4.9}$$

In our examples, we again use a geometric cooling schedule with a starting temperature of 5 dropping to a final temperature of 0.1 over 50 steps, unless otherwise stated. This is followed by a strictly downhill search starting from the image with the highest value of $\phi(x)$ during the simulated annealing stage.

Example 4.7. A STAGE 4 MOVE.
Figure 4-8 shows an example of a possible move for one vertex. The vertex is currently at the position labelled A. A new position, B, along that pixel edge is selected at random. Moving the vertex from A to B means increasing the proportion of black in the pixel on the left of the vertex by the area of the triangle ABC. Similarly the proportion of the pixel on the right is increased by the area of the triangle ABD. Therefore the likelihood needs to be recalculated for both pixels. The prior penalty for edge length around the boundary has increased by the difference between |CB| + |BD| and |CA| + |AD|. No other pixels are changed so their posterior probabilities cancel out when calculating the ratio in Equation (4.9). □

Remark.

- Since the number of vertices in the image during Stage 4, is usually much less than the number of pixels in the image, the optimisation in Stage 4 is not particularly

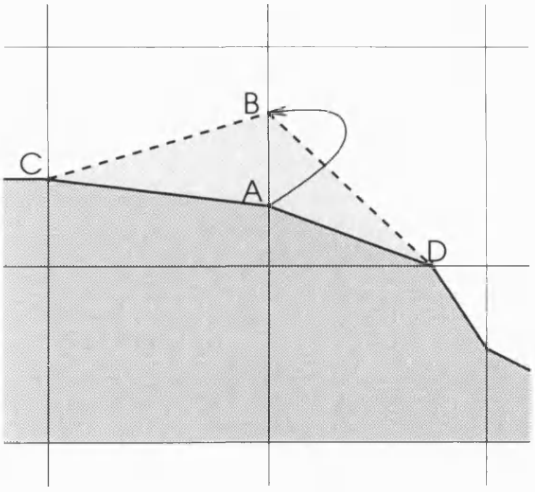

Figure 4-8: A STAGE 4 MOVE. The Metropolis algorithm proposes a move from the point A to a point B randomly selected along the common pixel boundary between the two linked edges. This move adds an additional edge length of $|CB| + |BD| - |CA| - |AD|$ and changes the proportion of black in the two pixels by the areas of triangles ABC and ABD. Equation (4.9) is used to accept or reject the proposed move.

onerous. This is in spite of the greater amount of computation required in each Metropolis update relative to the Gibbs sampler updates of Stages 1 and 2.

- Each move during this stage means updating the state of two pixels simultaneously. This is another example of multiple-site updating, the previous example occurred with the 2×2 block updating in the previous chapter.

Relationship to template matching

In Stage 4, each proposal consists of a uniformly distributed random displacement to a single vertex, affecting two boundary segment lengths and three inter-segment angles. This changes the boundary of the object and consequently both the likelihood and prior terms in the posterior. In this respect, this stage of the algorithm is rather similar to a simplified form of template modelling. However, no prior knowledge is assumed about the number or shape of the objects in the image and the displacement of a single vertex is constrained to move along a single pixel boundary only.

Rerouting the edges in Stage 4

In Stage 4, the vertices are free to move along the pixel edge; but not to move into neighbouring pixel edges, as this would require the addition of line segments. Thus, the dimension of the optimisation problem during Stage 4 is fixed at the start of that stage, so r does not change. The possibility of inserting new elements into a vector is excluded, for simplicity.

The parameter vector needed to define a reconstruction requires one variable to specify the location of each vertex around the boundary of each object. If we allow new links to be added to the boundary around an object, the target distribution does not have a density with respect to a simple measure in a parameter space of fixed dimension r. Instead, each rerouting adds or deletes some vertices from the reconstruction and this changes the

dimension of the parameter vector that is used to define a reconstruction. To allow such additional vertices to be added, the appropriate parameter spaces would then become a union of subspaces of varying dimensions.

Example 4.8. REROUTING EDGES DURING STAGE 4.
Consider moving a single vertex. Currently, it has to remain on the edge between the two

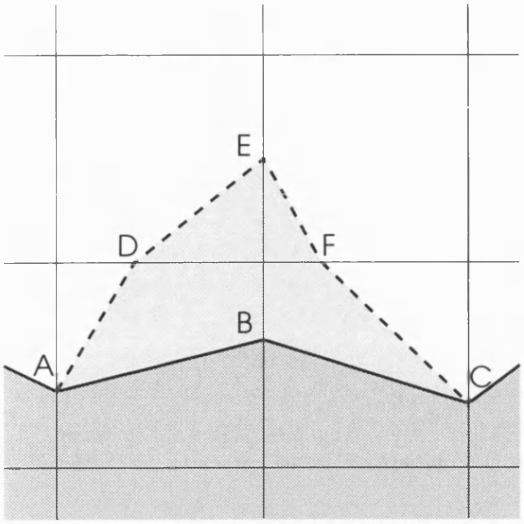

Figure 4-9: REROUTING EDGES DURING STAGE 4. The vertices A, B and C could be rerouted to become the sequence A, D, E, F and C. The number of vertices around an object increases by two. Finding the optimal position of the vertices for the new set of vertices means searching over a parameter space with an extra two dimensions.

pixels on which it initially lies. We could consider allowing it to move to another edge, effectively rerouting the boundary around the object.

If the current vertex jumps to a new pixel, the number of vertices, defining the boundary around an object, will increase or decrease by up to two vertices. For example, in Figure 4-9, the sequence of edges A, B and C becomes the sequence A, D, E, F and C. We are no longer simply trying to optimise the location of the vertices in the reconstruction, we are also trying to optimise the number of vertices that we should have. So our state space becomes a union of state spaces *à la* Green (1995c).

As with all our algorithms, we are ultimately aiming at a 3D model, so any 2D model must be easily extended to 3D. □

Jubb and Jennison (1991) describe an algorithm which allows edges to be re-routed through a new sequence of pixels in their equivalent of our Stage 4. Such re-routeing would certainly be desirable in simulating from the posterior distribution in Equation (4.6) over all of Ω.

Some novel MCMC algorithms have been developed to deal with this scenario by using deformable templates and jump-diffusion simulation (Grenander and Miller 1994; Srivastava, Miller, and Grenander 1991), which would allow regions of one colour to appear or disappear. Transitions between models of different orders are made at random exponential times and between jumps the parameters of the model are simulated to satisfy a stochastic differential equation. Phillips and Smith (1994) used this sampler to investigate a number of model choice problems. Green (1994a, 1995c) describes an explicit, more general class

of MCMC methods that use reversible Metropolis-Hastings jumps between subspaces that might also be applicable in this context.

4.4 An application in microscopy

In this section we illustrate how the subpixel model can be used to restore blurred microscopic images of the type considered by Hitchcock and Glasbey (1994). There are three examples, the latter two come from the same record.

The fungal mycelium *Trichoderma viride* consists of a network of hyphae from a single fungal organism. It is growing on a microscope slide which has been coated with a cellophane-coated nutrient agar (Ritz and Crawford 1990). A fundamental characteristic of this class of fungi is their mycelial growth form. It is an effective mechanism by which habitats can be explored, in order to find new food bases and subsequently exploit them. The fungi need to search their habitat efficiently to maximise their chances of finding food, simultaneously minimising the amount of energy consumed. Some evidence suggests that the fungi forage according to the spatial distribution of their food. Traditionally, results were based on qualitative data from field and laboratory experiments. Today, image analysis can be used to examine the spatial structure of the fungal hyphae in relation to their environment. Glasbey and Horgan (1994, Chapters 5 and 6) use a thinning operation to get a skeleton that is one pixel thick to estimate the total length of hyphae. They allow for effects due to lines being represented as lattice points rather than being in continuous space. Crawford, Ritz, and Young (1993) discuss further work on fungal morphology and its relationship to soil structure.

4.4.1 Fungus Mycelium 1/3: a simple data structure

In Plot (a) of Figure 4-10, we consider just a small and structurally simple example of a fungal growth, in order to show the effect of the model and algorithm in detail. At each of the 41 $\times$ 51 pixels, a blurred greyscale value is observed and the record value is recorded as an integer in the range 0 to 255. The image consists of two long fungus arms on the left and some isolated objects spread across the image. Several of these smaller objects may be only a few a pixels in size. Our objective is to remove blurring and noise, down to a subpixel level. Subsequent processing of the reconstruction can then lead to summary information about the fungi, which reflects their foraging or feeding behaviour as they spread across the microscope slide.

In this example, we briefly summarise how the parameter estimates are obtained. The next two examples consider the problems associated with parameter estimation in more detail.

The two image colour estimates

The foreground and background colours were estimated from a histogram (not shown) of the data to be 249 and 50 respectively.

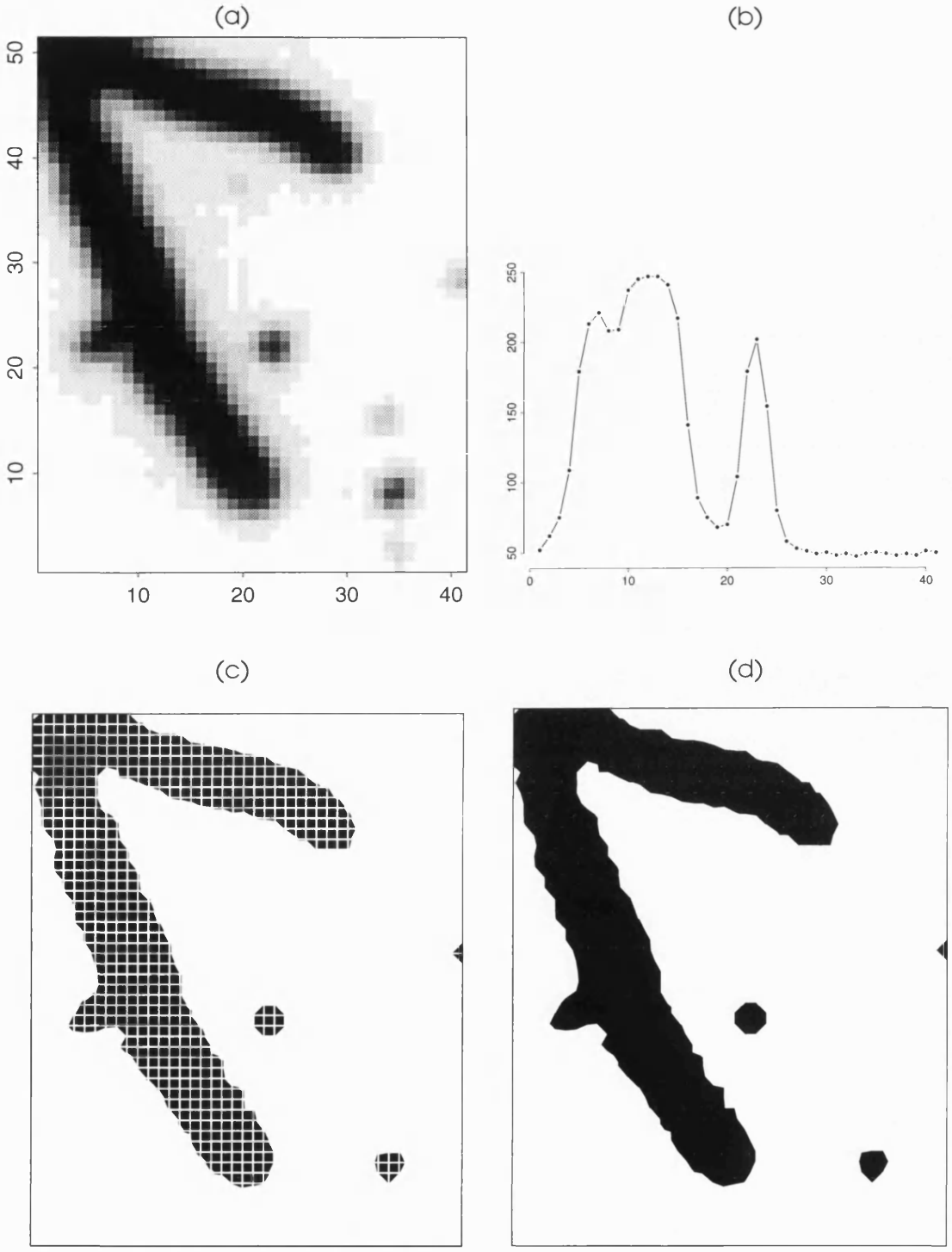

Figure 4-10: Fungus Mycelium Growing on a Slide — 1/2. Plot (a) shows a part of a fungus mycelium growing on a microscope slide. The blurring kernel is estimated from cross-sections of the image. Such a cross-section, along row 22 of the image, is shown in Plot (b). Plots (c) and (d) show a reconstruction of the data where the smoothing parameter is $\beta = 50$. The images are the same except that the left image has an artificial boundary imposed to highlight the individual pixels. The ragged object boundaries suggest that the smoothing parameter needs to be increased.

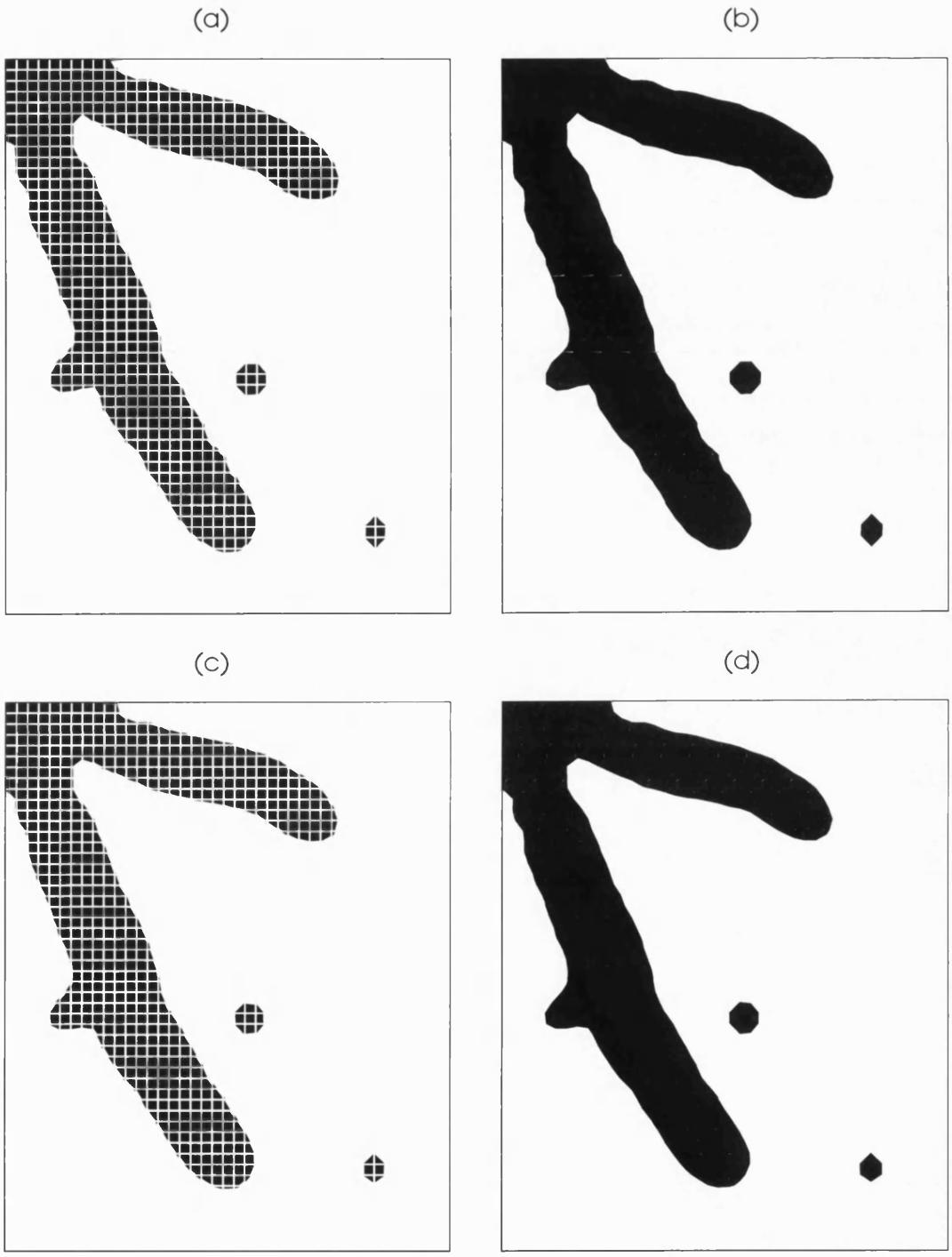

Figure 4-11: FUNGUS MYCELIUM GROWING ON A SLIDE — 2/2. Each row of this figure shows two reconstructions of the data. The value of the smoothing parameter in Plots (a) and (b) is 250 and in Plots (c) and (d) is 450. The choice of about $\beta = 250$ seems to best fit our priori knowledge about the true image containing smooth boundaries.

The noise estimate

The record variance, for Equation (4.5), is estimated to be 2.5, by sampling in parts of the image which contain just one colour.

The blurring kernel estimate

The blurring kernel is estimated by taking cross-sections through the data. If no blurring is present, the cross-section would be almost vertical at the boundary between the fungus and the background. The rate at which the cross-section moves from the one colour level to the other provides an estimate of the blurring kernel. The top-right image in Figure 4-10 shows the colour values for row 22 of the image. This row passes through one of the arms and an isolated object lying to the right. The remainder of the cross-section consists of the background colour. Repeating the analysis elsewhere in the image suggests that a Gaussian kernel with a standard deviation of about 1 is a reasonable approximation for the blurring. Hitchcock and Glasbey (1994) decided that a Cauchy distribution best approximates the blurring kernel but they also felt that a Gaussian distribution was an acceptable approximation.

The temperature schedule

As usual for Stages 1, 2 and 4, simulated annealing is used with a starting temperature of 5 and a final temperature of 0.1. During each of Stages 1, 2 and 4, one hundred sweeps of the image on a geometric schedule are applied, followed by ICM to convergence. Tests with a larger number of sweeps did not make a significant difference to the reconstruction. Other temperature schedules were also considered but the results were not sensitive to small changes in these parameters. The reconstruction with the lowest energy during each simulated annealing run is used as the starting point for a strictly downhill convergence to the nearest local minimum, to complete each stage.

Comments on the analysis

The bottom row in Figure 4-10 and the two rows of images in Figure 4-11 show reconstructions for increasing values of the smoothing parameter in Equation (4.6), $\beta = 50$, 250 and 450 respectively. For the left-hand image in each row, a pixel grid is superimposed and the region of black in each pixel has been drawn to lie *inside* the boundary of each pixel, to illustrate the amount of subpixel detail recovered.

The parameter β determines the balance between our desire for smooth boundaries, in the sense of minimising the edge length, while still agreeing with the observed image. If β is large, we move away from the record and objects tend to have smooth boundaries. Conversely if β is small then the observed image is closely matched but the boundary around objects is less smooth. As the value for the smoothing parameter increases, the boundaries become smoother and isolated objects are more likely to disappear.

Our prior expectations about the smooth boundaries around objects in the true image leads us to tentatively suggest that a smoothing parameter value of $\beta = 250$ is a suitable choice. Currently this parameter is chosen by trial and error. Obtaining a more objective, data-driven estimate is the focus of future work. Computer experiments suggest that the reconstruction is not overly sensitive to the choice of the smoothing parameter.

4.4.2 Fungus Mycelium 2/3: a complex data structure

In §4.4.1, the structure of the fungus in the image was deliberately chosen to be simple, to demonstrate the model and algorithm. In Figure 4-12 (a), a different fungal record is shown. It contains much more structure than the previous record but the analysis is similar. The purpose is to demonstrate that the algorithm makes no assumptions about the size shape or orientation of the object in the image.

Again, Figure 4-12 (a) is an image which is essentially binary in nature, with little noise and blurring. Unlike the previous example, the objects in the image, the fungal hyphae, are typically only 1–2 pixels wide so subpixel reconstruction may be more beneficial. The fungus is spread out across the image so as to maximise its ability to feed on the nutrient that lies on the microscope slide.

The two image colour estimates

Each pixel, in the image in Figure 4-12 (a), is recorded as an integer ranging from 0 to 255 (1 byte per pixel), as usual. This record has 40 unique values, ranging from 30 to 241. The histogram in Plot (b) counts the number of occurrences of each observed record value. A log scale is needed to display the data but the right y-axis scale shows the actual pixel count. The 40 unique record values are plotted along on the x-axis, where the height of each line corresponds to the number of occurrences of that record value. The most common record value is 30, the background colour. This occurred 20% $(789/(63 \times 60))$ of the time. The foreground colour, with a record value of about 240, is less obvious in Plot (b). Two reasons for this are: the fungus occupies a very small proportion of the area over which it is spread and the log scale of the y-axis. The former reason is to be expected because the fungi want to cover as much area as possible, to maximise their chances of finding food, but they also want to minimise the cost of doing so, since spreading out over an area means using energy to grow. A clearer picture can be deduced from Table 4.1.

Table 4.1 tabulates the values in the record shown in Figure 4-12 (a) and the linechart in Figure 4-12 (b). The number of occurrences of each of the 40 record values is shown. The intermediate values in the table may be due to a combination of noise, blurring and the grey-level colour in the background of Figure 4-13 (a). (The source of this grey-level colour is not known.) The figures suggest that the foreground colour is about 240 and the background colour is about 30. Plot (c) shows the record values, denoted by a circle, for the 30^{th} row and column of Plot (a). These are the horizontal and vertical lines that partition the image into quarters, approximately. Record values going from right to left along row 30 are drawn as a continuous line; record values going from top to bottom along

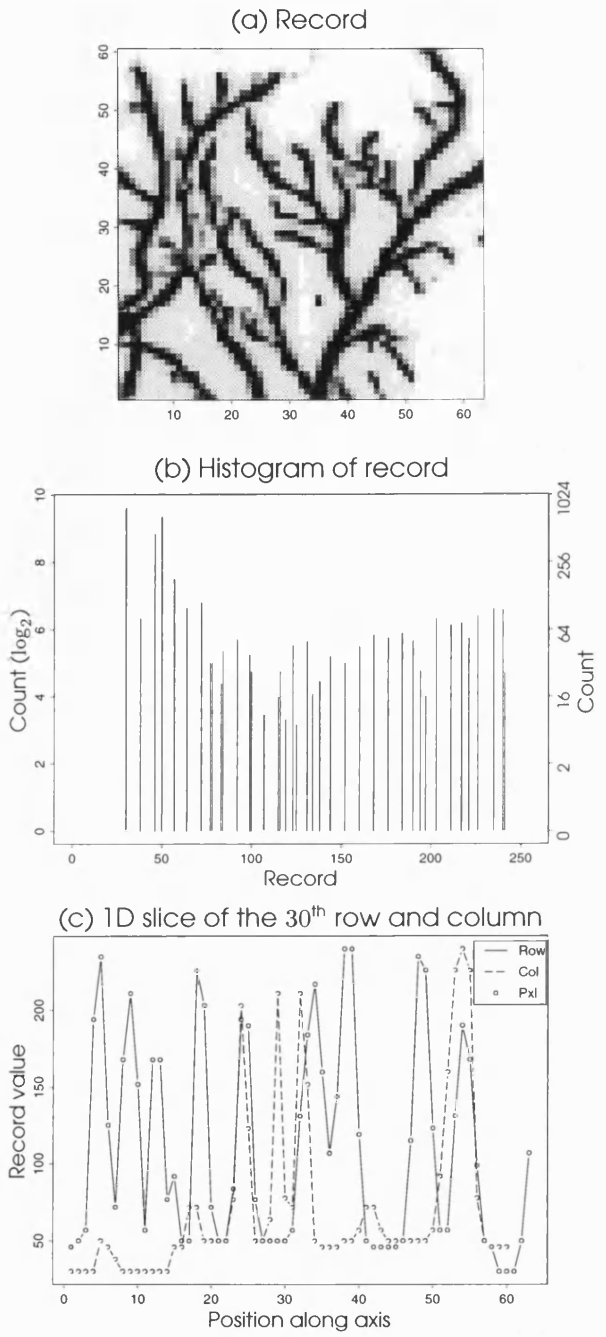

Figure 4-12: Complex Structure of the Fungus Mycelium — 1/2. Plot (a) shows a part of a fungus mycelium growing on a microscope slide. The image is 63 x 60 pixels in size. Plot (b) shows a histogram of the record values. Plot (c) shows the record values for the 30th row and column of the image in Plot (a). Plot (c) shows more clearly a foreground colour of about 240 and a background colour of about 5. The slope of the line in going from the foreground to the background colour suggests that only a small amount of blurring is present.

Record	30	38	46	50	57	64	72	77	78	83
Count	789	81	456	653	181	100	112	32	32	21
Record	84	92	99	100	107	115	116	119	123	125
Count	41	52	38	27	11	16	27	10	46	9
Record	131	134	138	144	152	160	168	176	184	190
Count	50	17	22	37	32	45	57	54	59	50
Record	194	197	203	211	217	221	226	235	240	241
Count	27	16	80	70	73	53	85	97	96	26

Table 4.1: RECORD COUNT FOR THE FUNGAL HYPHAE IN FIGURE 4-12. The 40 unique observed record values in the 63 x 60 image are shown. The number of occurrences of each record value is also tabulated. The figures suggest that the foreground colour is about 240 and the background colour is about 30.

column 30 are drawn as a jagged line. This also suggests a foreground colour of about 240 and a background colour of about 30. We now need to estimate the noise and any blurring.

The noise estimate

The noise is estimated by calculating the variance of record values from areas of a single colour. Figure 1-1 (a) (re-shown in Figure 4-14 (a) for convenience) shows the original image, from which Figure 4-12 (a) is taken. The empty regions in this image are used to find large areas of background colour only. This suggests a variance of about 4. For this image, it is only feasible to estimate the variance for the background colour because it is difficult to find large areas clearly composed of the foreground colour. Therefore, we assume that the variance of the foreground colours is similar.

The blurring kernel estimate

In Figure 4-12 (c), the slope of the line in going from the foreground to the background colour indicates the extent of the blurring. A vertical line with no pixels on it would suggest no blurring. Most of this line seem to be about 2–3 pixels wide, which suggests that there is a small amount of blurring. (In fact, if blurring is ignored then reconstructions have very jagged edges.) To allow for the blurring, a 3×3 blurring window with a Gaussian kernel and a variance of 1 was found to be quite successful.

The temperature schedule

Two hundred sweeps were used during Stages 1 and 2; four hundred were used in Stage 4. For each stage, the starting and ending temperatures were 5 and 0.1 respectively.

Comments on the analysis

Figure 4-13 (a) redisplays Figure 4-12 (a), for ease of comparison.

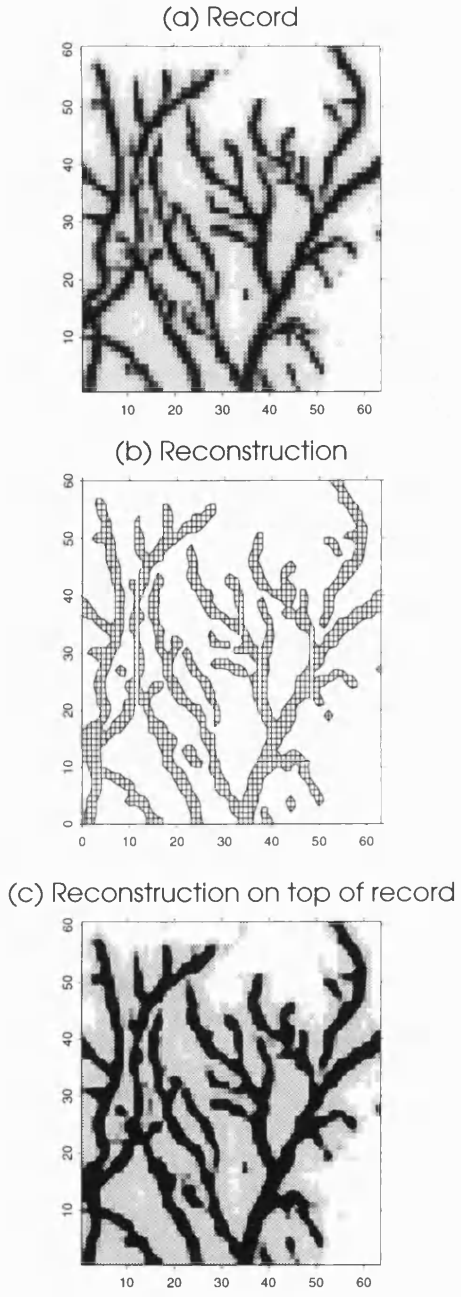

Figure 4-13: COMPLEX STRUCTURE OF THE FUNGUS MYCELIUM — 2/2. Plot (a) redisplays Figure 4-12 (a) for comparison. Plot (b) shows a reconstruction using Algorithm 4.1. A boundary is drawn around the region of black within each pixel. Clearly the algorithm is capable of dealing with highly irregular shapes of different sizes and orientations. Plot (c) consists of Plot (b) superimposed onto Plot (a) but with the foreground region coloured in black. This highlights the dark grey pixels that might be part of the fungus but which are excluded in this particular reconstruction.

Plot (b) shows a reconstruction from Algorithm 4.1, where a boundary has been drawn around the region of black in every pixel. A smoothing parameter of $\beta = 300$ is used. In this example, it is difficult to specify both the shape and the number of objects in the record. The subpixel algorithm provides an objective method for detecting the closed boundaries in this image down to a subpixel level. The analysis suggests that some of the fungal hyphae are only one or two pixels wide, so full pixel methods might not be adequate.

Plot (c) shows Plot (b) superimposed onto Plot (a), except that the regions of black within each pixel in Plot (b) have been filled in. Here, the purpose is to outline the regions identified as being part of the fungal network. In doing so, it highlights the small regions of dark grey pixels that are not part of the fungal mycelium, in this reconstruction. Sensitivity analysis is needed to decide if the parameters need adjusting so that such regions are also included in the reconstruction.

4.4.3 Fungus Mycelium 3/3: zooming in on detail

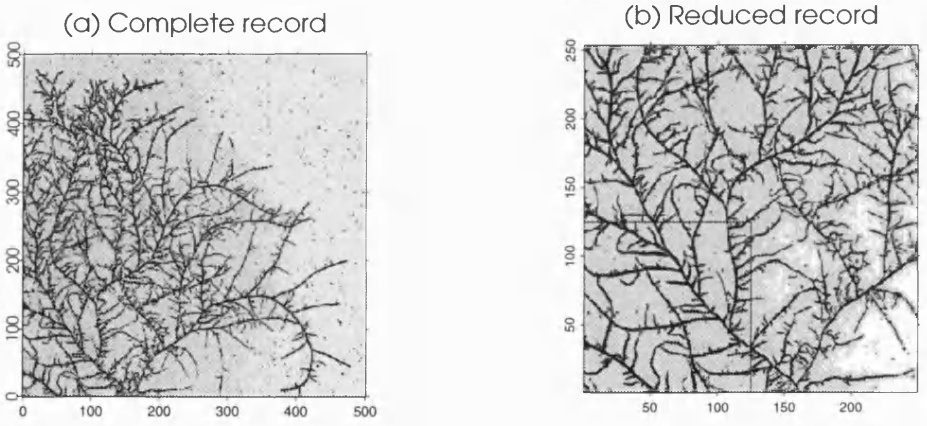

Figure 4-14: THE ORIGINAL FUNGAL MYCELIUM DATA. Plot (a) shows the full record, which is 500 × 500 in size. Most of the fungal growth is in the bottom left hand side of this image. For this reason, we only use the 250 × 250 block of pixels in the bottom left hand side. This record is shown in Plot (b). (The highlighted box in the lower left hand corner of Plot (b) is discussed later.)

The original record, used in this and in the previous section, is shown in Figure 4-14 (a). It is 500 × 500 in size but most of the detailed structure is in the bottom left hand corner and this sub-image is shown in Figure 4-14 (b).

Figure 4-14 (b) is a much larger record than the previous examples. To show that the algorithm works on large and complex structures, consider reconstructing the underlying fungus from the record shown in this image. Unlike the record analysed in §4.4.1, the records in §4.4.2 and §4.4.3 are taken from the same source. Therefore, we expect the parameter values in this section to be similar to those in the previous section.

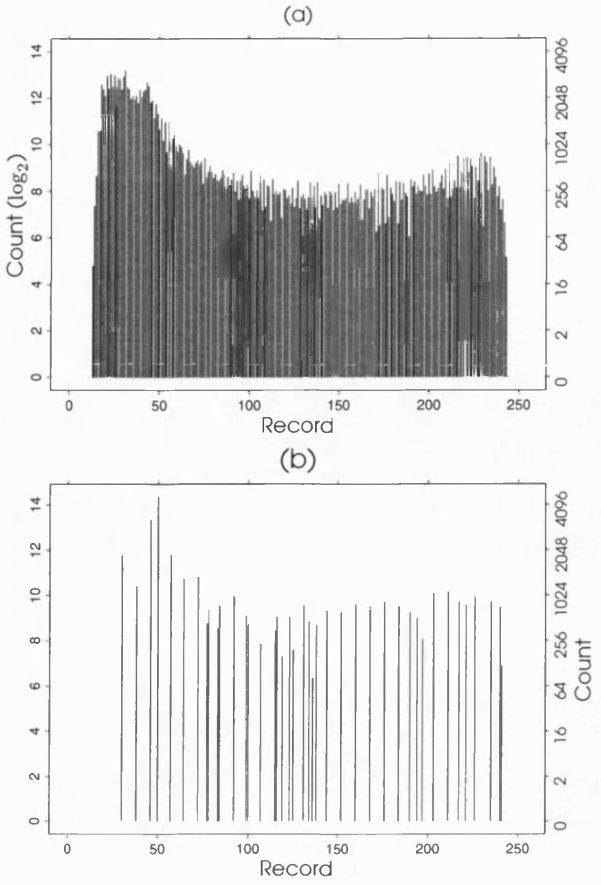

Figure 4-15: Histograms of the Original Fungus Mycelium Data. Plot (a) shows a histogram of the record values in the image shown in Figure 4-14 (a). Plot (b) shows the corresponding histogram for Figure 4-14 (b). A foreground colour of 240 is used. Plot (b) suggests a background colour of 50 but the histogram for the whole image suggests a value of about 30, which agrees with the analysis in §4.4.2.

The two image colour estimates

The histograms corresponding to the images in Figure 4-14 are shown in Figure 4-15. The foreground colour is estimated to be 240, as before. The estimate of the background colour is problematic. Plot (b) suggests a background colour of 50 rather than 30, as in the previous example. However, the histogram of the whole image (Plot (a)) suggests a background colour of 30, which agrees with the histogram in Figure 4-12 (b). In practice, the difference is not significant.

The noise estimate

The noise is estimated in the same way as before. Because this record is from the same source as the previous example, the variance is the same, $\sigma^2 = 4$.

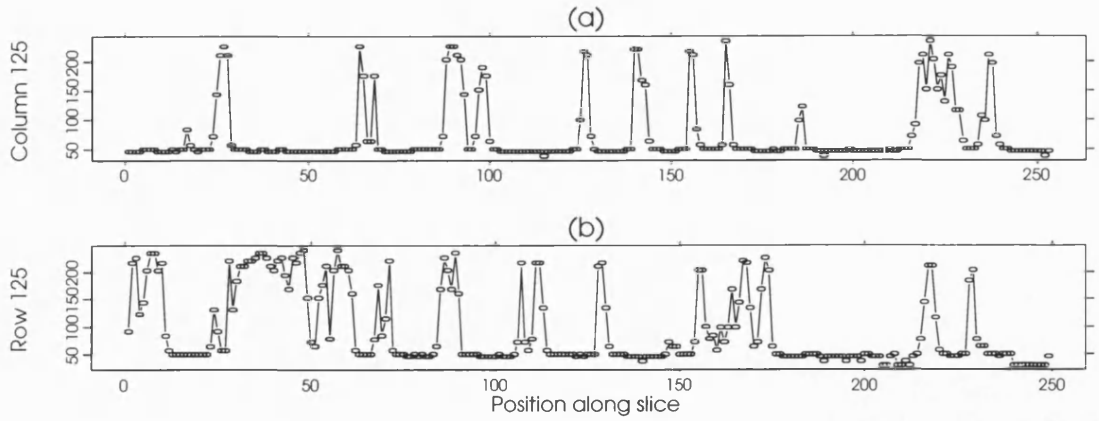

Figure 4-16: THE 126TH ROW AND COLUMN OF THE FUNGAL MYCELIUM DATA. This figure shows the record values for the 126th row and column of the image in Figure 4-14 (b). As in the previous section, this gives some indication of the extent of the blurring in the image and the values of the foreground and background colours.

The blurring kernel estimate

The one dimensional horizontal and vertical slices that partition Figure 4-14 (b) into quarters are shown in Figure 4-16. Following a similar analysis to the previous section, they can be used both to estimate the two colours and the extent of any blurring. A Gaussian blurring kernel with a variance of 1 is used, spread over a 3×3 window.

The temperature schedule

The starting and stopping temperatures of the geometric schedule are 5 and 0.1 and 50 sweeps of the simulated annealing algorithm are used in Stages 1 and 2 and 100 sweeps are used in Stage 4.

Comments on the analysis

A smoothing parameter of $\beta = 300$ is used, to ensure that all the parameters are consistent with the last example.

Once the reconstruction is complete, it retains most of the information of the original record; but it can be stored, retrieved and displayed at a fraction of the cost. (The reconstruction in Figure 4-17 (a) requires 17,079 edges; the record in Figure 4-17 (b) requires 253×249 pixels.) For example, to see detail in the reconstruction, we simply zoom in on the feature of interest. Figure 4-17 (c) shows the highlighted square box in the bottom left hand side of Figure 4-17 (a). It can be draw much more quickly than the corresponding record, shown in Plot (d).

Because the reconstruction is accurate to a subpixel level, the process of zooming can be carried further. The square box in Figure 4-17 (c) is re-shown on a larger scale in Figure 4-18 (a). The highlighted square in this reconstruction is shown in Figure 4-18 (c).

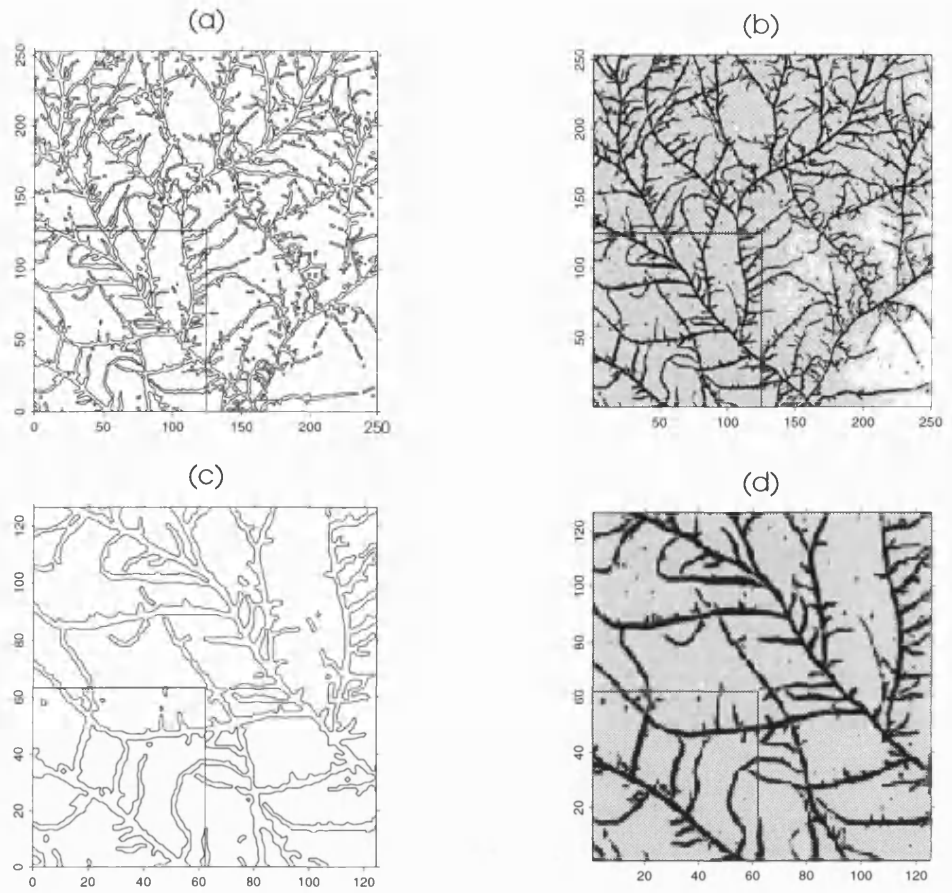

Figure 4-17: Zooming in on Detail in a Reconstruction — 1/2. Plot (a) shows a reconstruction of the image in Figure 4-14 (b). This is a demonstration of the model and algorithm on an image that is both large and complex. An important advantage of the algorithm is that it generates a reconstruction which contains most of the information of the corresponding record, shown in Plot (b); but it can be stored retrieved and displayed at a fraction of the cost. Plots (c) and (d) zoom in on the bottom, left-hand-side of Plots (a) and (b), respectively.

For each reconstruction, the corresponding record is shown immediately to the right. As the scale increases, visual inspection of the record does not become any easier. However, the detail in the reconstruction does become apparent down to a subpixel level.

Analysis of images to a subpixel accuracy is almost always carried out on images that are much smaller in size than the image in Figure 4-17 (a). Because of its size, the reconstruction takes 18 hours on a Sparc server 1000 and requires 27mb of RAM. Reducing the number of simulated annealing sweeps still produces similar estimates but at a fraction of the CPU time.

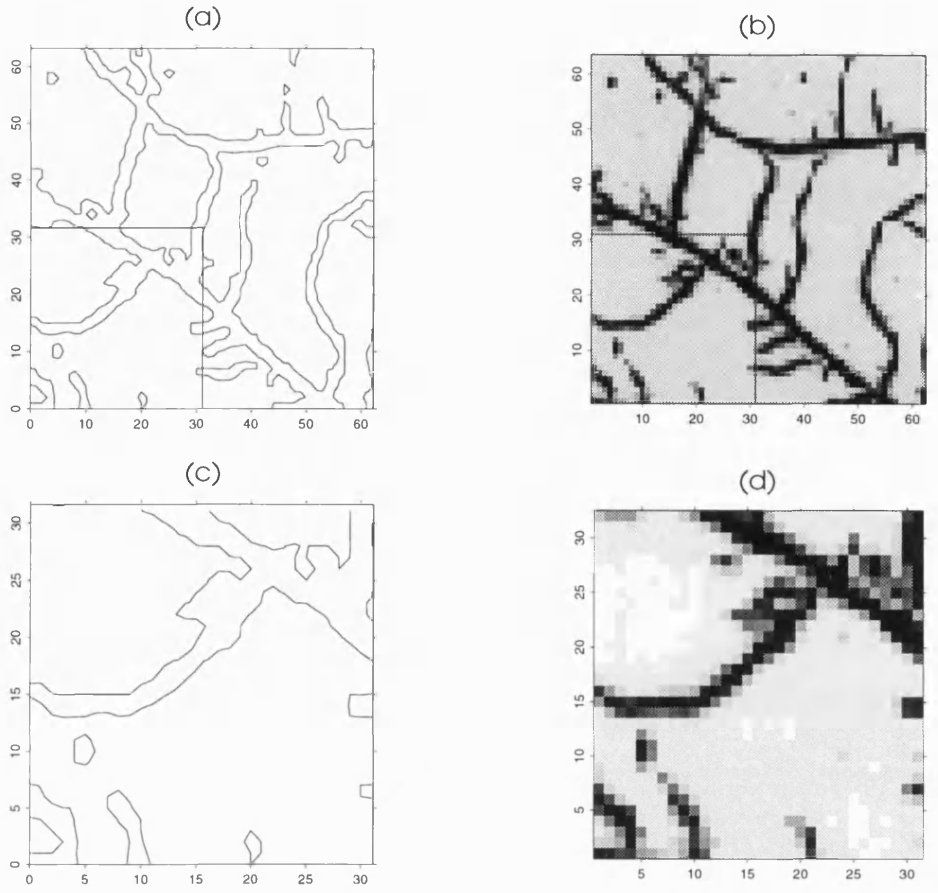

Figure 4-18: ZOOMING IN ON DETAIL IN A RECONSTRUCTION — 2/2. Plots (a) and (c) continue the trend in Figure 4-17 by zooming in on the bottom left hand side of Figure 4-17 (c) and Figure 4-18 (a), respectively. No computation is required to generate these two plots. They are simply a subset of Figure 4-17 (a). They show the detail of the information that is in the reconstruction. For comparison, the corresponding records are shown in Plots (b) and (d), respectively.

4.5 Final Remarks

A multi-stage algorithm is used to gradually refine the reconstructed image down to a subpixel level. The final estimate from one stage is used as the starting point for the next stage. This breaks a large optimisation problem down into a more manageable format. Furthermore, each stage in the reconstruction has a good starting point so the amount of work required at each stage is reduced. One advantage of this method is that the pixel grid on which the reconstruction takes place does not have to correspond to the resolution of the underlying record. For example, a coarser reconstruction grid may allow the records associated with each pixel to be averaged, increasing the signal-to-noise ratio but the cost is a poorer approximation to the data. One feature of this cascade approach is that parameter values need to be consistent from one stage to the next (Hurn and Jennison 1995). In Algorithm 4.1, the model and hence the parameter values are constant

in Stages 1, 2 and 4; it is the state space that changes. This is in contrast to the previous chapter, where the smoothing parameter required adjusting between levels.

The algorithm proposed here, Algorithm 4.1, is computationally intensive. It is mainly applicable to small objects that are only a few pixels in size, given current computer processing power. However, it also works for large images. As show in this section, the data can be analysed on a small part of the image first, perhaps using just ICM, to get some preliminary results before long runs are carried out on the whole image. If several images are similar, perhaps because they are from the same source, the same parameters can often be used on each image, without constantly reassessing them.

In this 2D algorithm, Stage 1 may be avoided because the Gibbs sampler is used in Stage 2 so that the two proposals in Stage 1 are always reconsidered with each Stage 2 update. This is not the case in the 3D algorithm so a separate Stage 1 is maintained here for consistency.

With a large image, several pilot runs are necessary in order establish the best parameter estimates. As yet, we do not have a data-dependent way of choosing the penalty for edge length β nor for choosing the rule to ensure that the boundary is piecewise continuous.

Conceivably, the posterior distribution in Equation (4.6) could be sampled rather than simply optimised (Besag, Green, Higdon, and Mengersen 1995, §6). This would lead to estimated confidence intervals for the posterior but this issue is not explored further here.

Some image reconstruction algorithms guarantee that the boundary around objects is closed by constraining the parameter space to be the set of closed boundary configurations. Thus any estimate of the optimal boundary will have the property of being closed by default. Helterbrand, Cressie, and Davidson (1994) argue that this ensures that the boundary is a one-pixel wide contour (i.e. pathological cases are avoided), the complexity of identifying the boundary is reduced because fewer configurations have to be considered and ambiguous boundaries are avoided. Our algorithm incorporates a specific stage, Stage 3, to ensure that the boundaries around all objects are closed and that ambiguous boundaries are avoided. This offers greater freedom in earlier stages to explore more general parameter spaces.

4.6 Summary of chapter

This chapter introduced a continuous, 2D subpixel model for edge detection.

- The benefits of continuous subpixel reconstruction are discussed and an overview of the method is outlined.
- The prior, likelihood and posterior distributions are defined. The parameters that need to be estimated in the model are
  - the two colours,
  - the noise,
  - the blurring kernel,

- the smoothing parameter.

No assumptions are made about the shape or number of the objects in the image. The temperature schedule must be decided but this is not critical. The algorithm does require iteration, so it is not fast. However, it does not rely on interpolation to get subpixel answers, as other authors have assumed. The limiting restriction is that the edge across a pixel is a straight line.

- A multi-stage algorithm is used to gradually refine the reconstructed image down to a subpixel level. The final reconstruction segments the pixels in the image by allowing boundaries to consist of piecewise-continuous, straight lines across pixels.

  The model and hence the parameters are constant in Stages 1, 2 and 4; it is the state space that changes from one stage to the next. Strictly speaking, Stage 1 is not essential but it is included here for consistency with the corresponding algorithm for three dimensions, where Stage 1 is essential.

  The algorithm is illustrated with a simulated example.

  The possibility of rerouting is briefly mentioned.

- An application is considered in detail. Three different images are considered.

  - The first is a small image, where the object has a simple structure. The branches are several pixels wide. It is discussed in detail to illustrate how the model and algorithm are applied in practice.
  - The second image is of a similar size but has much more structure. So it is more challenging for the algorithm.
  - The final example is of a large image. Such reconstructions make it possible to zoom in on the features of interest and to store and display the information more easily.

- Finally, some overall comments are made.

Chapter 5

Continuous 3D subvoxel reconstruction

Visualisation is a necessary part of data analysis.
Tools matter.
W. S. Cleveland
Visualising Data

An approximate answer to the right problem
is worth a good deal more than
an exact answer to an approximate problem.
John W. Tukey.

5.1 Introduction

5.1.1 Background to the algorithm

In this chapter, we present a methodology for extracting a topologically closed set of objects from a volume data set. This technique is of growing importance because of the advent of non-destructive sensing equipment, especially confocal microscopes, which generates point samples of true, three-dimensional (3D) objects.

The confocal microscope

Shaw and Rawlins (1991) suggest that there are three reasons for the current popularity of light microscopy. The ease and convenience of optical microscopy make it possible to examine much larger numbers of specimens than is possible using electron microscopy. Secondly, recently developed labelling techniques, especially immunofluorescence microscopy, allow almost any cellular component to be specifically imaged because labelling dyes improve the contrast in the image, unlike conventional light microscopes. Thirdly, optical microscopy is a relatively non-invasive and non-destructive technique and this makes it possible to observe living cells.

The imaging characteristics of a conventional microscope are complicated. The resolution in the direction of the optical axis (denoted by z) contains out-of-focus contributions from other parts of the specimen above and below the plane of focus, so the depth of field is larger than the in-plane resolution limit. Consequently, the image at a given focal plane is a poor estimate of a true section through the specimen. A confocal laser scanning mi-

croscope (CLSM) is essentially a conventional scanning microscope with a point detector where the light source has been replaced by a laser. The main advantage of CLSM is its sectioning property which offers a modification to the optical system to improve the rejection of out-of-focus components (Howard 1990). In particular, it can visualise a thin slice within a thick specimen without having to slice it mechanically. In a conventional scanning microscope, it is only possible to see a projection of the whole object. The confocal microscope is the device used to record the data which are analysed in this chapter but the method proposed here can be applied to data from other recording devices.

Visualisation in 3D

Our model and algorithm are the three dimensional analogue of the model and algorithm in Chapter 4. Images are binary in nature, consisting of a 'black' object set against a 'white' background. Pixels are replaced by voxels, edge length is replaced by surface area in the prior and area is replaced by voxel volume in the likelihood.

A 3D object may be visualised in a number of ways. Some hardware has the ability to make parts of a 3D image transparent. However, this facility is not available on Sun workstations. So we represent individual record elements by voxels. An all black *voxel* contains 6 faces, 12 edges and 8 vertices. It is the 3D analogue of the all black pixel in 2D, which has 4 edges and 4 vertices. When an image is visualised, voxels which are deemed to lie outside the object are coloured white and all voxels inside an object are coloured black. If a voxel overlaps a boundary then only that part of the voxel that lies inside the object is coloured black.

Example 5.1. EXAMPLE OF A 3D IMAGE FROM A CONFOCAL MICROSCOPE.
Figure 5-1 shows an example of a 3D visualisation, created by thresholding the record. The record measures vacuolar volumes in stomatal guard cells of *Commelina communis* stained with acridine orange, recorded using a confocal microscope. Optical sections were collected at 1 μm intervals. The hole in the middle of a guard cell closes up when the two halves of the cell contract. Measurements were repeated at 20 minute intervals during opening and closing responses. The difference between the volume and surface area as the cell switches from being open to being closed is what concerns plant scientists at Oxford, who supplied the data (Fricker and White 1992). They seek estimates of quantities, such as the surface-to-volume ratio.

One single optical section near the centre of the guard cell is shown in Figure 1-1 (b), on page 2. The reconstruction shown in Figure 5-1 is $13 \times 17 \times 16$ in size and it is formed by simply thresholding the record at a sensible value. In this image, the guard cell is in the open position so that it is approximately toroidal in shape. Notice that surrounding the guard cell, there is a small isolated object only a few voxels in size.

As a first approximation to the volume, we can simply count the number of full-voxels inside the object, as shown in Figure 5-1. Similarly, we can count the number of faces lying between white and black voxels to estimate the surface area but Howard (1990) points out that tiling algorithms will invariably underestimate surface area, sometimes severely.

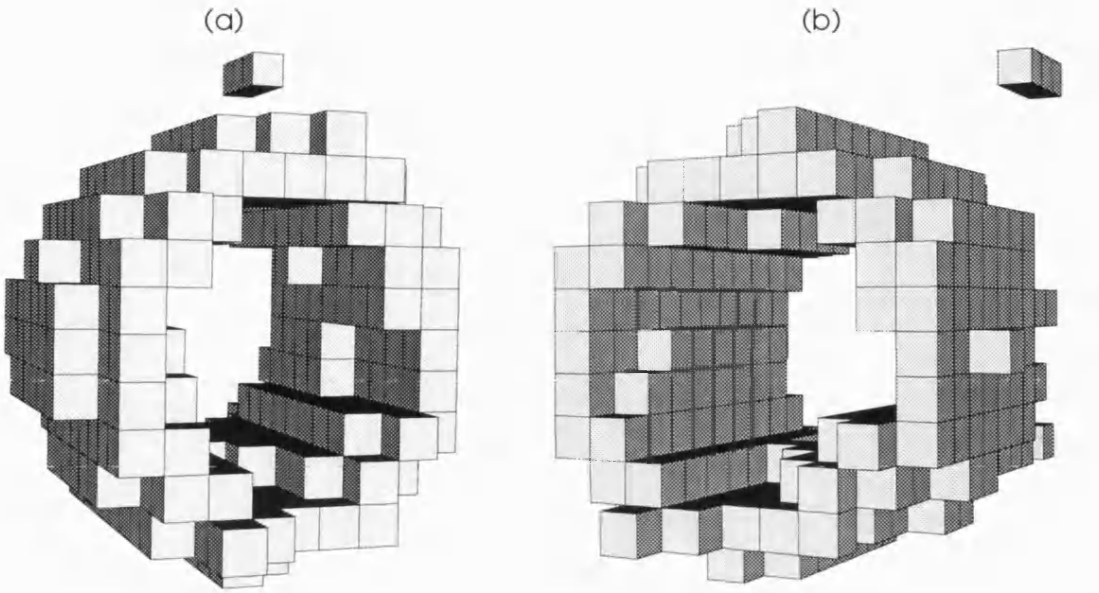

Figure 5-1: EXAMPLE OF A 3D IMAGE FROM A CONFOCAL MICROSCOPE. This 3D reconstruction shows a confocal microscopic image of a single plant cell, called the guard-cell. (One 2D slice of the underlying data set is shown in Figure 1-1 (b) on page 2.) The camera is shifted to the left in Plot (a) and to the right in Plot (b) to improve the depth information. Individual record elements are represented by voxels, which are made visible if that record element is inside the object. An isolated voxel is evident in the upper right-hand corner.

The subvoxel method developed in this chapter reduces the effect of the discrete grid upon which the data are recorded. □

Before discussing our 3D model and algorithm in detail, we briefly review existing 3D visualising algorithms.

5.1.2 Existing visualisation methods

Segmentation in 3D is a major objective of 3D imaging, just as in 2D. Non-destructive examination of the interior of an object is now possible using remote scanning and sensing technology. So a number of research groups and commercial organisations are currently developing image processing algorithms to analyse directly the resulting complicated data sets. The data sets may consist of the interior of a mechanical object or a patient's internal organ. Current applications are mainly in medicine and engineering (see the citations in this section for more details).

Statistical image analysis offers new methods for reducing the systematic errors present in any 3D data set and enhancing the clarity and contrast of relevant features. To date, these methods has been limited to the visual integration of data and their representation to the human observer; but it is possible to perform some measurements on these 3D images as well.

An important characteristic of this field is the huge volume of information that is generated by 3D recording devices. This makes it difficult to extract useful information about a physical object from a 3D data set, especially as the corresponding software tools are not yet completely developed. Also, each type of object seems to require a specific set of processing algorithms and parameters, as in 2D. The following algorithms are among the most commonly used at the moment:

Stacking 2D slices

A three dimensional object can be digitised, and hence visualised, by physically taking a regular, ordered sequence of cross sections through it. For example, confocal microscopy and magnetic resonance imaging devices collect 3D arrays of data by generating a three-dimensional scalar field, where each scalar records some physical property at that point, such as density. Often, the observed data consist of 2D slices stacked vertically to form a 3D volume.

The usual practice in the past was to analyse 2D slices separately and 3D information was then inferred from the results. (The raw, 2D data can be analysed using some of the 2D techniques mentioned in Chapter 1, such as is done in morphology (Serra 1982).) Then the data are visualised by displaying an ordered, sequence of 2D slices through the image, in quick succession.

Simply displaying successive 2D slices is an intuitive and simple way to visualise a 3D data set. Alas, it is usually not good enough because the human brain is unable to extract structural information from a stack of 2D slices. Also, there may be a focal plane outside of which parts of the object become blurred.

Stereology

This is the fusion of two images whose parallax reveals depth information, especially if the object is opaque. The two images are formed by viewing the object from two slightly different angles. As with the 'stacking' solution above, two dimensional image analysis techniques can then be applied to the captured image. Again, there may be a focal plane outside of which parts of the object may be blurred.

Traditionally, 3D imaging consisted of stereological interpretation of conventional two dimensional images to obtain information about the three dimensional structure which they sample.

Rendering algorithms

A much favoured route to 3D data analysis is to create a projection of the 3D image or a subset of it. Adding motion, by rotating the view, creates the illusion of three-dimensionality. There are two commonly used classes of rendering algorithms: volume rendering and 3D graphics.

Volume rendering Reconstructions are formed by colouring each element of the record and projecting the semi-transparent voxel onto a chosen projection plane. There are three steps in volume rendering: classification, shading and projection.

Classification This step assigns an opacity value between 0 and 1 to each voxel. Typically, the lower threshold value is selected to eliminate background intensity. The range between lower and upper threshold controls the degree of transparency in the final reconstruction.

Shading Shading techniques simulate both surface characteristics and the position and orientation of surfaces with respect to light surfaces and the viewer.

Projection Next the semi-transparent, coloured record element is projected onto a plane perpendicular to the observer's direction. A ray is cast from each point on the projection plane through the record. For every record element on its path, the colour and opacity is accumulated, in back-to-front order, to produce a single pixel colour.

Because all the record elements contribute to the reconstruction, even badly defined features are present. This allows complex images to be displayed.

3D graphics Another approach is to create a geometric model of the object using the volume data as a guide. Such a geometric model can be used to analyse and interpret the data in a number of ways: object recognition, inspecting or visualising the model and calculating geometric measurements, such as volume.

In these algorithms, the surface of the true image is approximated by a list of polygons. It is displayed by projecting all the polygons onto a plane that is perpendicular to the viewing direction. Most modern graphic workstations offer built in 3D graphic functions, which can render polygons at speeds of several thousand to several million per second. The view of the reconstruction is changed by applying a rotation matrix. Transparency and lighting models can enhance the view further.

Although the rendering algorithms are now very sophisticated, it is still difficult to generate a list of polygons that accurately represents the surface of the reconstruction. Miller et al. (1991) suggest several approaches for extracting 3D geometries from volume data: manual model building, contour stitching, surface construction, marching cubes and deformable primitive models.

Manual model building Interactive model building is possible if a display system is available that allows users to select model points using an on-screen cursor to point at the record. The procedure is simply to connect all the vertices that are to be linked together to form the 3D structural model. The display system ideally needs to be able to rotate the data and to display the record simultaneously on different display screens. Displays shown might include cross-sections, volume rendering and orthogonal views.

Contour stitching This method combines a sequence of 2D contours together by fitting a triangular strop between adjacent contours. Lin and Cline (1989) use a similar approach but use splines to do the stitching. Both cases require the contours within each slice to be identified and problems arise if there are different numbers of contour lines in adjacent slices.

Marching cubes One of the most popular methods is to triangulate a 3D surface through the volume data. Lorensen and Cline (1987) use what they call 'marching-cubes' to generate a list of polygons, from volume data which contains no connectivity information.

Each voxel in their reconstruction is bounded by eight vertices, consisting of four neighbouring vertices lying on each of two adjacent 2D slices. (Note that this definition of a voxel is *not* the same as that which we use in the rest of this chapter.) A record element from the volume data is associated with each of the eight vertices or corners of the voxel. Then each corner is classified as lying inside or outside the object by comparing its value with a threshold or contour level, chosen by the user. The voxel is then triangulated so that each triangle forms a portion of the surface. The exact location of the vertices of the triangulation is found by *interpolation.* Connectivity is automatically generated because the four corners on the face of any voxel are the same as the four corners on the opposing face of the neighbouring voxel.

Cline et al. (1988) extend marching cubes to an algorithm called 'dividing cubes', by resampling the voxels to a different level of resolution.

Surface construction Kass, Witkin, and Terzopoulos (1987) use 'snakes' to model contours by minimising a spline function. The function that is minimised is based on the image and its first and second derivatives. This model can use prior information but the minimisation of the spline is a global operation that may lead to computational difficulties.

Deforming primitive models Finally, Miller et al. (1991) deform a primitive model to fit an object using local geometric constraints. The reconstruction is a closed, topologically-simple (non-self intersecting), geometric model of an object. They envision the reconstruction as the surface of a balloon placed inside an object and expanded until its surface reaches the boundary of the scanned object.

The vertices of this mesh of polygons are deformed using local information and different levels of resolution are possible. The reconstruction is closed at all stages. However, the algorithm may result in self-intersecting polygons and the initial placement of the primitive may be important. This method approximates the observed data by aggregating the observed data as opposed to analysing the observed data directly. It can model non-convex objects and the level of detail can be varied to allow for quick estimates. The amount of computation

is proportional to the size and complexity of the object, not the size of the original data.

Overall, the 3D graphics models often assume only one object in the image and results are sensitive to the size and shape of the object, in some cases. Generally, only local solutions are found but this does mean that less computation is required.

The paper by Chen, Swedlow, Sedat, and Agard (1995) offers an introduction and further references to rendering algorithms.

5.2 The model

5.2.1 Background

In our model voxels are indexed by $i = 1, \ldots, n$. A surface may pass through any voxel, splitting it into two regions of different colour. We allow such a surface to consist of a sequence of triangles linked by the sharing of a common edge. This is the 3D analogy of using a straight line to separate two regions of colour in a pixel. In the same spirit as the previous chapter, we allow at most a *single, triangulated surface* to divide any voxel between the two colours in our reconstructions and we incorporate this property into our prior model for the true, continuous scene.

A *triangulated surface* means a linked sequence of triangles, with no holes. An *internal face* refers to a single triangle on the surface that separates the two regions of colour within a voxel. An *external face* refers to that part of one of the six faces of a voxel that is coloured black. These definitions will be clarified in later examples.

5.2.2 The prior distribution

In order to specify a prior penalty, we first define an image space, Ω. For the 3D problem, we define Ω to be on a class of binary images in which boundaries are continuous and piecewise planar. The object is coloured black and the background is coloured white, denoted by b and w, respectively.

Definition 5.1. We define the *image space* Ω to be the set of binary images satisfying the conditions:

- Each 3D image X consists of an ordered, 3D array of n voxels, $X = \{X_i : i = 1, \ldots, n\} \in \Omega$.
- Each voxel is either a single colour or divided into two regions of different colours by a single surface composed of one, two or three triangular sections.
- A face of a voxel with a single colour must lie next to a face of another voxel with a single colour, excluding boundary voxels.

- Each surface through a voxel shares a common edge with surfaces through the adjacent voxels that the surface borders. There are three or four neighbouring voxels, except at the image boundary.

Denote the colouring of voxel i by $h_i(x)$, as before. We also use the notation $x(z)$ to denote the colour of an image x at the point $z \in \Re^3$, in the true continuous image.

The neighbourhood around a voxel in 3D could consist of 6, 18 or 26 voxels, if a voxel were connected to those other voxels with which it shares a common face, edge or vertex, respectively. To be consistent with the previous chapter, we use a 'first' order neighbourhood only in defining a MRF model for the prior image distribution. So the neighbourhood for a voxel is the set of *six* voxels with which it shares a common face. Consequently, the colouring for a voxel is conditionally independent of all other voxels, given the colouring of these six neighbouring voxels.

A MRF based on *surface area* defines a prior distribution for the scene X. Surface area is the sum of the areas of the triangles separating regions of different colours.

Model 5.1. PRIOR. *The prior distribution for an image is based on the total surface area between black and white in that image, $S(X)$. We define the prior probability for the scene X to be*

$$f_X(x) \propto \exp\{-U_X(x)\} \stackrel{\text{def}}{=} \exp\{-\beta S(x)\} = \exp\left\{-\beta \sum_{i=1}^{n} S_i(x)\right\}, \quad x \in \Omega, \tag{5.1}$$

where $S_i(x)$ is the surface area in voxel i. □

This model is consistent with the Hammersley-Clifford theorem for the general form of a MRF, given in Equation (2.3) and the prior p.d.f.s used in the last two chapters. The density in Equation (5.1) is defined with respect to a measure in a similar way to §4.2.1.

The motivation for this choice of prior is similar to the motivation for the 2D prior in Chapter 4. Now, instead of edge length across pixels, we use the surface area passing through voxels. For the image space Ω, this surface area is the sum of the areas of the triangles that pass through voxels. Lower values of the smoothing parameter β lead to a greater probability of objects in the image having long, usually jagged, triangular surfaces.

5.2.3 The likelihood function

The observed data $Y = (y_1, \ldots, y_n)$ are recorded on a regular lattice of points $\{z_i \in \Re^3 : i = 1, 2, \ldots, n\}$. The sensor's output at each Y_i represents the average intensity within that voxel and is assumed to be proportional to the volume occupied by the object in that voxel. We denote the average value of the colour of voxel i by

$$h_i(x) = bp_i^b + wp_i^w, \tag{5.2}$$

where p_i^b and p_i^w are the proportions of pixel i covered by colours b and w respectively,

$$p_i^b = \iiint_{z \in i} I_{[x(z)=b]} dz \quad \text{and} \quad p_i^w = \iiint_{z \in i} I_{[x(z)=w]} dz.$$

(The algorithmic details for calculating the volume of a colour inside a voxel are deferred until §5.3.7 on page 130.)

Due to imperfections in the recording sensor, the record Y may be degraded by additive noise. We assume an additive Gaussian model, without blurring, for the record. If necessary, the model can be generalised to include blurring, in a similar manner to that discussed in the previous chapter.

Model 5.2. LIKELIHOOD. *The likelihood model is*

$$y_i = h_i(x) + e_i \tag{5.3}$$

where e_i is independent, additive sensor noise, with distribution $N(0, \sigma^2)$. □

As before, we assume the noise variance σ^2 is known or can be estimated from the data. From Equation (5.2), the distribution of the signal y given the image x is then

$$f_{Y|X}(y, x) \propto \exp\Big\{-(2\sigma^2)^{-1} \sum_{i=1}^{n} \big(y_i - h_i(x)\big)^2\Big\}. \tag{5.4}$$

5.2.4 The posterior distribution

As usual, the posterior is the combination of the prior and likelihood, which in this model is

$$f_{X|Y}(x, y) \propto \exp\Big\{-\beta S(x) - (2\sigma^2)^{-1} \sum_{i=1}^{n} \big(y_i - h(x_i)\big)^2\Big\}. \tag{5.5}$$

We want to find the image x in the sample space Ω which maximises the posterior probability $f_{X|Y}(x, y)$ defined by Equation (5.5), the MAP estimate.

5.3 The algorithm

5.3.1 An overview

The 3D reconstruction algorithm contains four stages, just like the 2D algorithm in the previous chapter. Our ultimate goal is to find the image x in the sample space Ω which maximises the posterior probability $f_{X|Y}(x, y)$ defined by Equation (5.5). The sample space Ω, as defined in §5.2.2, is much more complex than in the 2D model. To get a good starting point in Ω, we again proceed in stages, optimising the *same* function over

different, simpler image spaces during Stages 1, 2 and 4 in order to obtain a good starting image in Ω for the final optimisation. At each stage we seek the maximum of

$$\phi(x) = \exp\{-\beta S(x) - (2\sigma^2)^{-1}\|y - h(x)\|^2\} \tag{5.6}$$

over images x in specified spaces, where $S(x)$ is the total surface area in image x and $h_i(x)$ is the average value of x over pixel i.

Note that there is consistency here with the previous two chapters: the objective function is fixed and we maximise it over a series of related image spaces. The answer from each stage forms a starting point for the subsequent stage.

- Stage 1 allows each voxel to be all black or all white, $\Omega_1 = \{x : x_i \in \{b, w\}\}$. Relative to the other stages, Stage 1 effectively makes large jumps across the image space Ω.

  During Stage 1 of the algorithm, there are no internal surfaces because all voxels are either all black or all white. Consequently, the penalty for two neighbouring voxels is the area of opposing colours on the common face lying between the two voxels.

- Stage 2 introduces surfaces through voxels but the vertices that define the internal faces of the surface through a voxel are constrained to lie midway along the voxel edge upon which each vertex lies. In addition, the internal surfaces do not necessarily link up with internal surfaces in neighbouring voxels, during Stage 2.

  When updating each voxel, the number of choices for the location of a surface is still very large, with several hundred possibilities from which to choose. This is in contrast with the 2D algorithm. A focussed design-strategy helps to reduce the number of possibilities in a sensible but easily-implemented manner. Consequently, all of the possible updates for a single voxel are not considered simultaneously.

- After Stage 2, a deterministic third stage is needed to guarantee that surfaces inside voxels link up with surfaces in neighbouring voxels, with as little disruption as possible.

- Overall, Stages 1 to 3 are intended to produce a good starting image $x \in \Omega$, for the final stage. In Stage 4, the vertices that define the internal faces of the surface through a voxel are allowed to move freely along the voxel edge upon which the vertex initially lies.

  There is no surface area on the external faces of any voxel in this stage because they are removed during Stage 3. The only prior penalty in Stage 4 is from the sum of the areas of the internal faces.

The relationship between the image spaces, corresponding to each stage of the algorithm, is still the same as in Chapter 4: $\Omega_1 \subset \Omega_2$, $\Omega_2 \supset \Omega_3$ and $\Omega_3 \subset \Omega_4 \equiv \Omega$.

5.3.2 The 3D continuous subvoxel algorithm

The algorithm is broadly similar to the 2D case (see §4.1 on page 73).

Algorithm 5.1. CONTINUOUS 3D SUBVOXEL.

Stage 1: Full pixel reconstruction *Define Ω_1 to be the set of images in which each voxel is of a single colour, all black or all white. From a convenient starting point, search for the image in Ω_1 that maximises $\phi(x)$, as defined in Equation (5.6), using simulated annealing based on the Gibbs sampler (see §2.3.3 on page 29).*

Stage 2: Initial subvoxel estimation *Define Ω_2 to be the image space where each voxel has:*

- *At most one internal surface passing through the voxel,*
- *The internal surface of the voxel is composed of one, two or three triangular sections.*
- *The vertices of any internal surface are constrained to lie midway along an edge.*

Starting from the final reconstruction from Stage 1, search for the image in Ω_2 that maximises $\phi(x)$.

Remark.

- There are no constraints forcing the surfaces in adjacent voxels to 'link together'.
- Because the space Ω_2 is so large, the algorithm used to generate proposals when updating a voxel is more complex than in 2D. In particular, the proposals we use for updating a single voxel are not symmetric. The details are presented in §5.3.5.

Stage 3: Conversion to an image in Ω *Apply a deterministic algorithm to convert the end product of Stage 2 to an image in Ω, with as little modification as possible (see §5.3.6 on page 126).*

Stage 4: Final subvoxel estimation *Starting from the reconstruction obtained in Stage 3, search for the image in Ω that maximises $\phi(x)$. This time a form of simulated annealing based on the Metropolis algorithm is used to move the vertex at which two inner faces meet along a voxel edge (see §5.3.7 on page 127).*

Remark. An implicit multiple update occurs during this stage. When the surface vertex shared by four voxels is moved, all four voxels are updated simultaneously. □

5.3.3 The Stage 1 reconstruction

In comparison to later stages, Stage 1 is relatively straightforward. There are just two choices for updating a single voxel: an all white or an all black voxel. Voxels do not have

an internal surface so the prior penalty $S(x)$ depends on the surface area of the external faces which differ in colour to the corresponding faces in neighbouring voxels. To find the maximum value of $\phi(x)$, as defined in Equation (5.6), the Gibbs sampler is used within the simulated annealing algorithm and this is followed by a strictly-uphill algorithm.

Example 5.2. STAGE 1 RECONSTRUCTION OF A 3D OBJECT.
Consider a reconstruction of the confocal microscopy example shown in Figure 5-1. Figure 5-2 shows a plot of the number of occurrences of each of the 237 unique record values which range from 11 to 250. This histogram is drawn on a log scale, with the original values shown on the right-hand axis. As expected, most of the image consists of the background colour. This plot and other analysis suggests a foreground colour of about 150 and a background colour of about 20. The peak for the foreground colour is less pronounced than that for the background colour. One reason for this may be the variation in record elements lying inside the guard cell, perhaps reflecting different parts of the cell structure. The plot suggests that the data could be thresholded at a value of about 100. The results of such thresholding are shown in Figure 5-1. The variance is estimated at about 16.

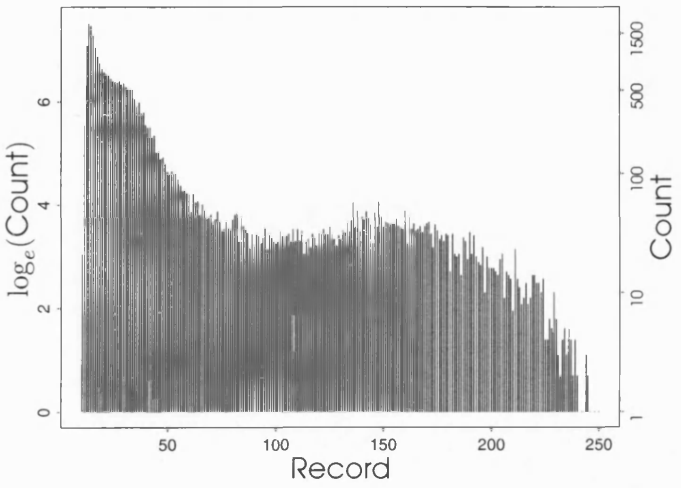

Figure 5-2: HISTOGRAM OF THE 3D CONFOCAL MICROSCOPE IMAGE. As expected, most of the image consists of the background colour. This suggests a foreground colour of about 150 and a background colour of about 20.

The Stage 1 reconstruction from this record is shown in Figure 5-3. A smoothing parameter of $\beta = 100$ is used. For Stage 1, the number of simulated annealing sweeps is 25 and the temperature drops from 5 to 2 before ICM is applied to convergence.

Stage 1 has a smoothing effect, as can be seen by comparing the Stage 1 reconstruction in Figure 5-3 and the thresholded image in Figure 5-1. There is one small object lying above and to the right of the guard cell which contains enough voxels to survive Stage 1. □

5.3.4 The Stage 2 image space

It is important to choose an image space and design a set of updating rules so that a rich class of reconstructions is generated. Recall from the previous chapter that we only allow

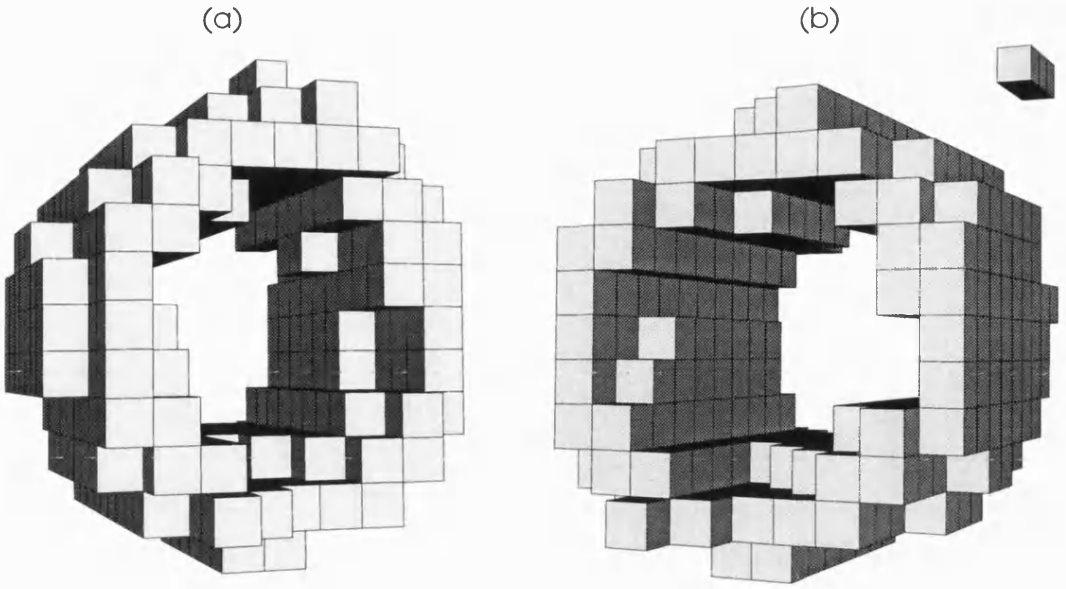

Figure 5-3: STAGE 1 RECONSTRUCTION OF A 3D OBJECT. For the original thresholded data shown in Figure 5-1, this figure shows a reconstruction of the guard cell at the end of Stage 1. Again, the camera is shifted to the left in Plot (a) and to the right in Plot (b) to improve the depth information. This stage smoothes the surfaces in the image.

one straight-line edge across a pixel in 2D, for simplicity. In the same spirit, we restrict the 3D image space in Stage 2 to the set Ω_2, defined in Algorithm 5.1. A voxel of an image in Ω_2 can be created by applying the following steps:

- Allocate a colour to each corner of a voxel.
- Construct each face of the voxel by introducing a vertex midway along each voxel edge that joins corners of opposite colours.
- Form an internal surface, if necessary, by combining triangular sections.

Some combinations of corner colours are not allowed. In particular, if an internal surface is present, it separates two regions of colour within the voxel and the surface consists of one, two or three triangular sections. There are sometimes a number of different ways to triangulate the internal surface, so separate cases need to be recognised.

This description provides a useful representation of a voxel, colour each corner to triangulate the internal surface. In particular, it eliminates voxels which contain more than one internal surface, to make the calculations more tractable. Nevertheless, the resulting image space is still rich enough to generate a wide range of reconstructions.

The set of Stage 2 voxels

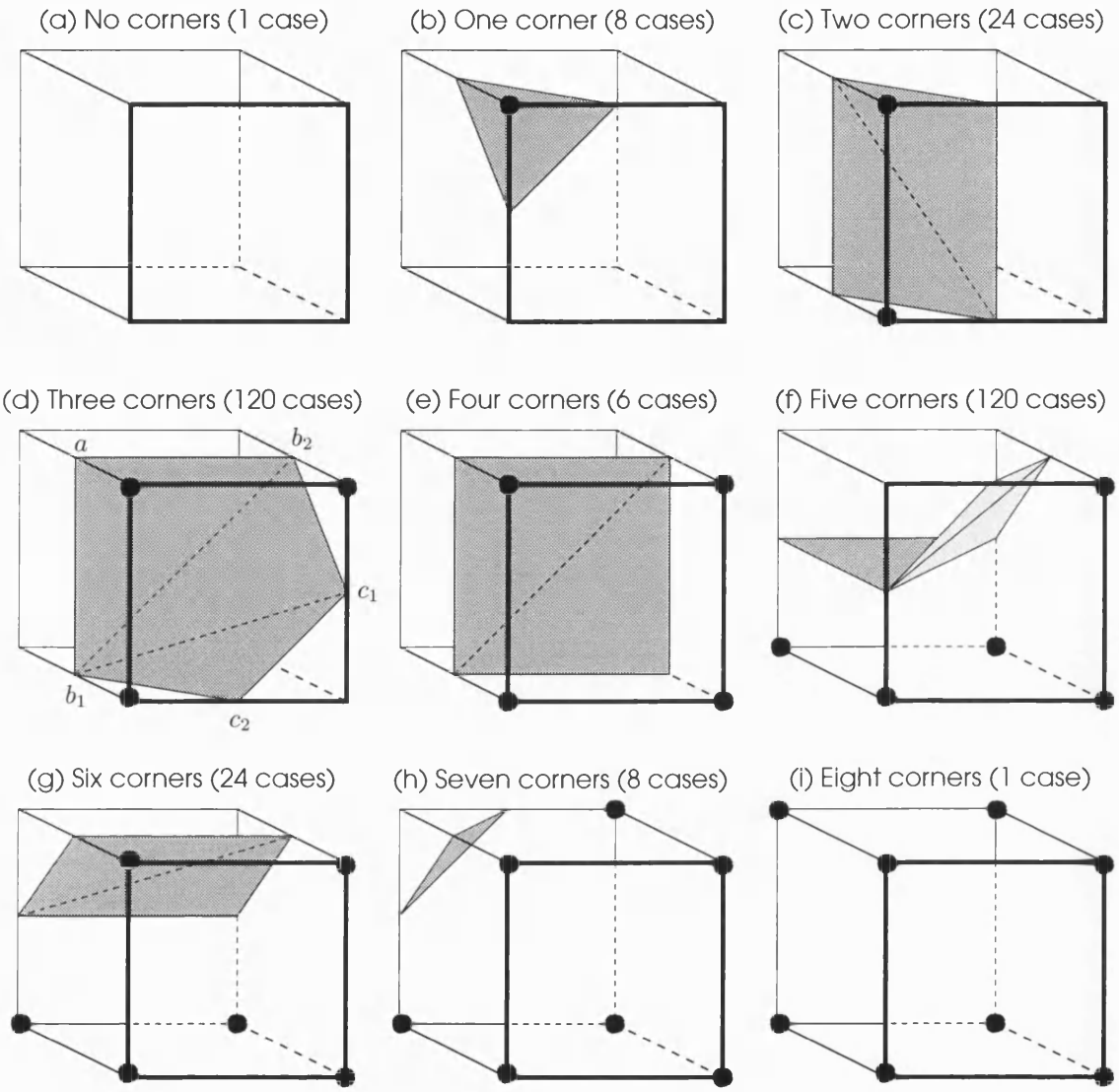

Figure 5-4: THE SET OF STAGE 2 VOXELS. This figure shows some examples of voxels in Ω_2 where the number of black corners increases from zero to eight in voxels (a) to (i) respectively. Each black corner is labelled by a black circle. For each voxel, the number of valid, unique voxels with the same number of black corners is shown in brackets. Some voxels have internal faces which are coloured to improve visibility. For each of voxels (c), (e) and (g), there are two ways the internal face can be constructed from two triangles but the internal face stays the same. For voxels (d) and (f), the internal surface is represented by three triangles, which have one vertex, the *apex*, in common. Voxel (d) shows the apex at b_1. There are five choices, a, b_1, b_2, c_1 and c_2, for the location of the apex which result in three distinct surfaces and volumes for that voxel because b_1 is identical to b_2 and c_1 to c_2.

Figure 5-4 shows all of the legal voxels in Stage 2, except for the possibilities that arise from rotation. A *legal* voxel is any voxel which in allowed in the image space Ω_2. The number of black corners increases from zero to eight in voxels (a) to (i), respectively. We refer to these voxels as *i–corner voxels*, where i can vary from 0 to 8. Each black corner in a voxel is labelled with a black circle and a bounding box has been drawn around each voxel to highlight its shape (most of the other examples do not have these features). The 0, 1, 2 and 3–corner voxels are the compliment of the 8, 7, 6 and 5–corner voxels, respectively.

Some of the voxels in Figure 5-4 have internal faces. These are composed of linked triangles. We refer to them as *internal faces*, as opposed to any black region on one of the six sides of the voxel, which is called an *external face*. The number of internal faces depends on the number of black corners in the voxel. The internal faces in Figure 5-4 are coloured to improve visibility. We allow at most three internal faces, arising from both 3–corner and 5–corner voxels. In principle, more cases could be considered but the extra complexity is unlikely to offer greater benefits.

For each voxel, the number of unique, legal voxels with the same number of black corners, allowing for rotation, is shown in brackets. For example, voxel (c) has two black corners which are separated by a single edge. Rotating this voxel on the x–y, x–z and y–z axes results in four different voxels for each of the three orientations. For each of these twelve possibilities, the internal surface (a rectangle) can be triangulated in one of two ways, depending on how the diagonal line across the rectangle is located. This distinction is not important during Stage 2 but is important in Stage 4. This results in a total of 24 possibilities for the 2–corner case.

Note that there are different ways to triangulate internal surfaces. All can be defined by specifying an 'apex'. The most complicated cases are the 3–corner and 5–corner voxels. One is the compliment of the other. The three internal faces all share a single vertex, the apex. The are five choices, a, b_1, b_2, c_1 and c_2, for the location of the apex which result in three *distinct* surface areas and volumes for that voxel because b_1 is identical to b_2 and c_1 to c_2. Figure 5-4 (d) shows the apex at b_1. For each voxel in Ω_2, there are a total of 312 possible updates for a single voxel.

To recap, a voxel is represented by a set of black corners, one of which is defined to be the apex, but there are some restrictions:

- For a 2–corner voxel, the black corners must linked by an edge.
- For a 3–corner voxel, one of the black corners must be linked to each of the other two black corners by an edge.
- For a 4–corner voxel, each of the black corners must be linked to two other black corners by an edge. This voxel looks like a cuboid.

The restrictions on 5–corner and 6–corner voxels are analogous to those for 3–corner and 2–corner voxels, respectively. We use this representation when deciding updates as we optimise over the image space.

The apex of a voxel

To help with the triangulation of the inner faces we define one of the vertices to be the apex.

Definition 5.2. APEX OF A VOXEL. For every permissible voxel, one and only one of the vertices in a voxel is defined to be the *apex*. The strategy for choosing the apex of a voxel depends on the number of internal faces in the voxel.

$$\text{Number of internal faces} = \begin{cases} 0 & \text{choose any vertex as the apex,} \\ 1 & \text{choose any of the three vertices on the internal face as the apex,} \\ 2 & \text{choose either of the two vertices shared by the two internal faces as the apex,} \\ 3 & \text{the apex is the vertex shared by the three internal faces.} \end{cases}$$

The apex is used to help classify the different kinds of Stage 2 and Stage 4 proposals and to calculate the volume of black in a voxel.

Example 5.3. CHOOSING THE APEX.
If the apex occurs at either of the vertices labelled b_1 and b_2 in Figure 5-4 (d) then the voxel has the same surface area and volume. Thus its contribution to the prior and likelihood penalties is the same. Similarly for the vertices labelled c_1 and c_2. So there are effectively only three unique choices for the apex. A similar argument applies to the 5–corner voxel. The apex of a voxel is chosen randomly because there is no extra information available to help make a choice, at this stage. □

Excluded voxels

Despite the 312 potential choices when generating a proposal during Stage 2, there are still many voxels which are excluded. We specifically exclude voxels where two or more separate internal surfaces are needed and voxels where the internal surface requires more than three internal triangles. This is in keeping with the simplifications made in the 2D model.

Example 5.4. SOME ILLEGAL VOXELS.
For example, the 3–corner voxel must have corners that are linked by an edge. If this were not the case then there would be more than one surface through the voxel. There are many other exclusions which are less obvious. Figure 5-5 shows some examples of *illegal*, 4–corner voxels. Plots (a) and (b) are illegal because it is not possible to triangulate a *single* surface through those voxels. Plots (c) and (d) are illegal because more than three internal triangular faces would be needed to create a surface separating the black corners from the white corners. □

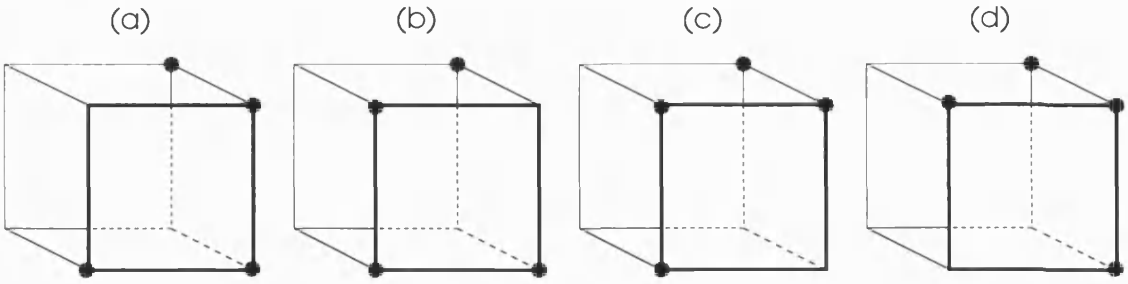

Figure 5-5: EXAMPLES OF ILLEGAL VOXELS. Several potential voxel reconstructions are specifically excluded. For voxels with four black corners, this figure shows some of excluded cases. These cases are excluded usually because of ambiguity over the choice of internal faces, such as voxels (a) and (b). Voxels (c) and (d) have more than three internal faces.

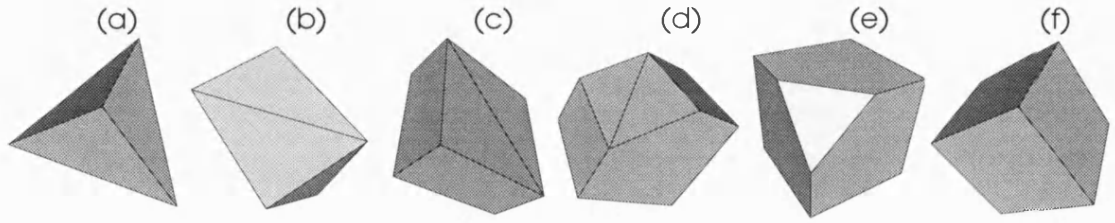

Figure 5-6: SOME STAGE 2 VOXELS. Shown here are computer-generated examples of voxels with $1, 2, 4, 6, 7$ and 8 black corners, in Plots (a) to (f) respectively. The voxels have been rotated and scaled to offer the best view. For example, Plot (a) shows a voxel with one black corner, pointing towards the viewer. It forms a tetrahedron. Each of the Plots (b)—(d) have two inner faces. The edge that splits the inner surface into two triangles can be drawn in two ways but the shape does not change.

Figure 5-6 shows actual examples of Stage 2 voxels. The colouring in the figure comes from flat-shading the shadows cast by a single light of medium intensity. Edges have been drawn around each face to aid visualisation. Each proposal has been rotated to give the best angle and scaled to fill the plot. So the only distinguishing feature between the plots is the shape of the black region in the voxel.

Looking from left to right, the plots in this figure show 1, 2, 4, 6, 7 and 8–corner proposals, respectively. For example, Plot (e) shows a voxel with seven black corners. The missing black corner, nearest to the viewer, is replaced by an inner surface, consisting of a single triangle. Each of Plots (b), (c) and (d) has two inner faces. The edge that splits the inner surface into two triangles can be drawn in two ways but the shape does not change. (However, this is not the case for the analogous voxel in Stage 4, where vertices are allowed to move freely.)

Figure 5-7 shows examples of the different types of 3–corner and 5–corner voxels that are shown in Figure 5-4 (d). In the first row, Plots (a), (b) and (c) each show one example of a 3–corner type-a, 3–corner type-b_1 and 3–corner type-c_1 voxel, respectively. Plots (d), (e) and (f) show the corresponding 5–corner voxels. These three classifications for 3–corner

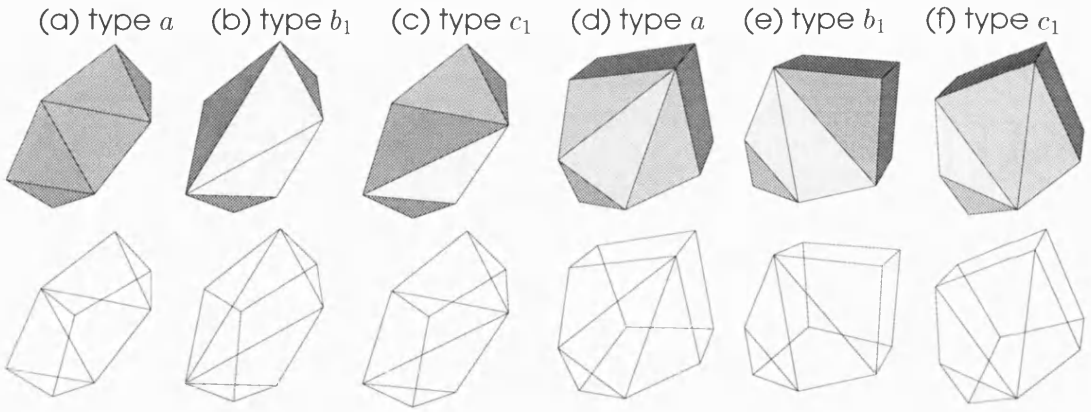

Figure 5-7: More Examples of Real Stage 2 Voxels. The first row shows three computer-generated examples each of 3-corner voxels, (a)—(c), and 5-corner voxels, (d)—(e). For each voxel, the second rows shows the same voxel but with the faces removed. There are three classifications for 3-corner and 5-corner voxels, which are labelled in Figure 5-4 (d). The first three columns show respectively examples of the 3-corner, type-a, type-b_1 and type-c_1 voxels first mentioned in Figure 5-4. The last three columns show the same classification for the 5-corner voxel.

and 5–corner voxels follow the labelling in Figure 5-4 (d). The image directly below each voxel, shows that same voxel but with the faces removed. So only the voxel edges are visible in the second row, to help to discern the shape of the voxels in the first row.

Summary of the image space in Stage 2

	Number of black corners								
Number of	0	1	2	3	4	5	6	7	8
Vertices	0	4	6	8	8	10	10	10	8
Edges	0	6	10	14	13	17	16	15	12
Faces	0	4	6	8	7	9	8	7	6
Cases	1	8	24	120	6	120	24	8	1

Table 5.1: Summary of the Set of Stage 2 Voxels. This table shows the number of vertices, edges, faces and combinations, for voxels with different numbers of black corners. The combinations arise from rotation of the voxel and permutation of the apex, if there are internal faces. For example, if two corners are coloured black then such a voxel would be described using 6 vertices, linked by 10 edges to form 6 faces. By rotation, there are 12 unique voxels with 2 black corners. There are four vertices at which to locate the apex but this reduces to just two ways of locating the diagonal on the internal surface, giving 24 cases in total.

The number of vertices, edges and faces needed to construct an i–corner voxel in Stage 2 depends on the number of black corners present in the voxel. These figures are tabulated in Table 5.1. This approach to voxel construction is reasonably efficient as most of the voxels in any reconstruction are all white, for which little information is needed. Further details about how a voxel is represented are outlined in Appendix B.3 on page 156.

In total, Table 5.1 shows that there are 312 potential choices when generating a proposal during Stage 2. (This figure exceeds the 2^8 possibilities mentioned earlier due to the different ways of triangulating the internal surface.) It is disappointing that this number is so large relative to the 2D algorithm, despite having placed restrictions on the kind of internal surface allowed and the location of the vertices of the internal faces.

The prior distribution during Stage 2

During Stage 2 the prior distribution of each voxel depends on the surface area of both the internal faces and the external faces which differ in colour to the corresponding faces in neighbouring voxels.

Example 5.5. THE PRIOR DISTRIBUTION DURING STAGE 2.
Let i, j and k index the spatial position of a voxel on the x, y and z axes, respectively.

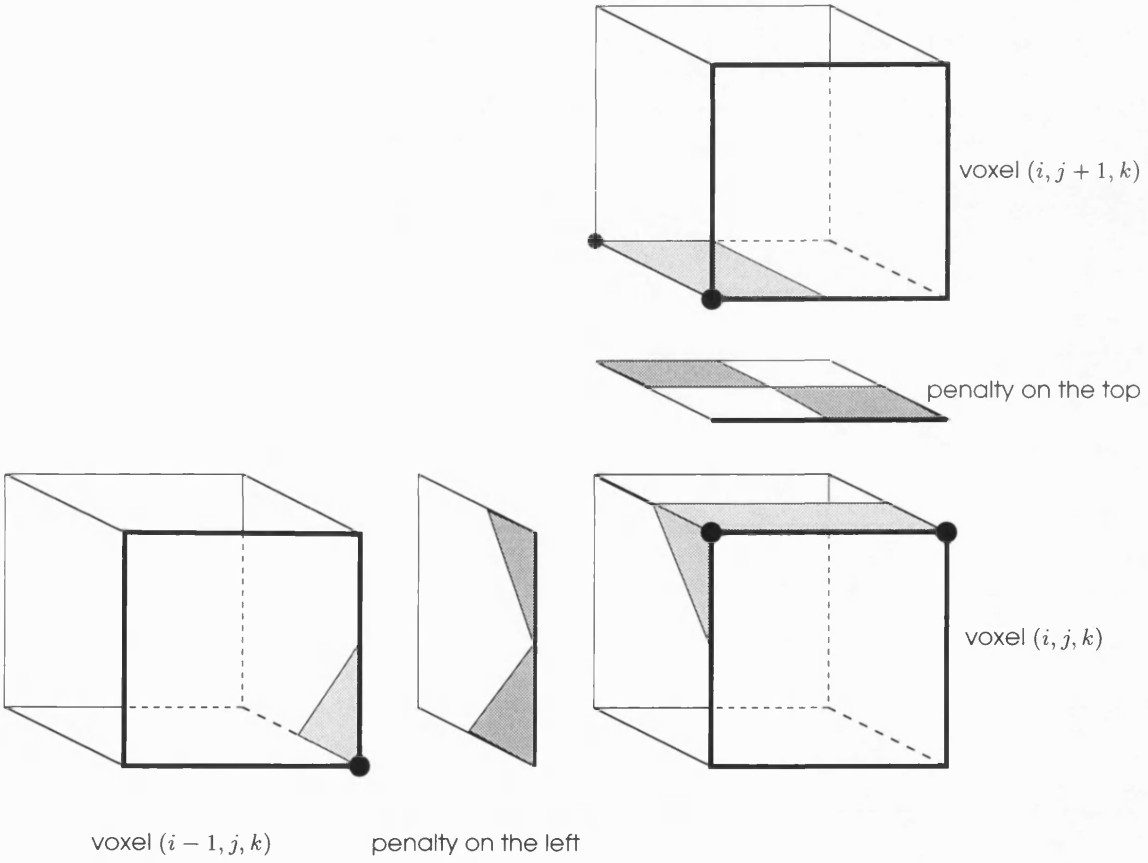

Figure 5-8: THE PRIOR DISTRIBUTION DURING STAGE 2. The prior penalties between voxel (i, j, k) and two of its six neighbours are shown. The penalty is the surface area which is coloured differently on two common faces. The shaded areas in the 'penalty on the top' and the 'penalty on the left' slices represent areas where the colours are not the same.

Figure 5-8 shows an example of the contribution to the prior penalty between voxel (i, j, k) and two of its six neighbours. The illustrated neighbours are voxel $(i, j+1, k)$ to the

north and voxel $(i-1,j,k)$ to the east. The penalty between the top face of voxel (i,j,k) and the bottom face of voxel $(i,j+1,k)$ is labelled 'penalty on the top'. The penalty between the left face of voxel (i,j,k) and the right face of voxel $(i-1,j,k)$ is labelled 'penalty on the left'. (Note that all other faces, including the interior of the voxels, have been omitted for simplicity.) The shaded area in these two slices represents area where the colours are not the same. The prior penalty, arising from these two of the six neighbouring voxels, is proportional to the sum of the shaded areas. □

5.3.5 Generating proposals for MCMC methods in Stage 2

The search method

We want to search for the image in the space Ω_2 that maximises $\phi(x)$ in Equation 5.6. There are many ways of designing the updating procedure in Stage 2. Stage 2 proposals cannot be generated using the Gibbs sampler because the amount of computation required is prohibitive. The Metropolis algorithm would also be costly to evaluate and the rejection rate is likely to be high due to the large number of choices. As always the key to our algorithm is to work with the corners of the voxel. Suppose a move is restricted to flipping the colour of just one of the eight corners in the voxel at a time. Then the proposed move depends on the number of black corners in the current voxel and the resulting similarity between the current state of the voxel and the new proposal means that the new proposal has a better chance of being accepted. The Hastings algorithm, defined on page 27, is used to choose the corner to flip and whether that move should be accepted or rejected.

However, the cost of this approach is that traversing the image space may be much slower. Instead of jumping from a 1–corner voxel to a 4–corner voxel, the algorithm has to move through at least one 2–corner and one 3–corner voxel. In particular, it is not possible to get from an all black to an all white voxel without making many moves. However, if such a large move were desirable then it would have been made during Stage 1. So the good starting point provided by Stage 1 reduces the amount of exploration needed in Stage 2. (The amount of exploration is explicitly controlled by the number of sweeps and the temperature during the simulated annealing algorithm.)

Operators for updating

While several different strategies were considered, an updating procedure that relies on a small set of simple rules is clearly preferable. To flip (change) the colour of just one of the corners in a voxel, we first define two operators, referred to as the *add operator* and the *delete operator.*

Definition 5.3. ADD OPERATOR. Flip the colour of a randomly chosen *white* corner which has the *greatest* number of black neighbours.

Definition 5.4. DELETE OPERATOR. Flip the colour of a randomly chosen *black* corner which has the *least* number of black neighbours.

With both operators, it may also be necessary to choose a vertex as the apex. There are several advantages to using an updating procedure that relies on these two simple rules.

- These two operators are simple to understand and to implement, yet the resulting image space is still very large.
- The list of proposals for updating a single voxel can be generated independently of the rest of the image. This localises the algorithm.
- *The two operators are rotation invariant regardless of the number of black corners in the current voxel.* This point is crucial because it means that the same code is used to generate an update for *any* voxel.
- Applying the operators to any legal voxel results in a legal voxel.
- All of the proposals generated are similar to the current state of the voxel and this increases the chances of the proposals being accepted, a desirable design feature. That is, the operators do not move the black region inside a voxel around too much *in a single move* but the state space is large enough to allow such a possibility as a result of a series of updates. So we can get from a voxel with any number of black corners at any orientation to any other legal voxel, though it may take several moves to achieve this. Thus, a reducible Markov chain is avoided because it is possible to get from any image to other image.

In order to further enrich the class of possible reconstructions, we consider choosing a proposal which contains the same number of black corners as the current voxel. This is done by applying the add operator immediately followed by the delete operator or vice-versa. (This includes the redundant possibility of not changing the voxel.) We refer to these two moves as *add-delete* and *delete-add operators*, respectively. The four operators add, delete, add-delete or delete-add are chosen with equal probability. The chosen operator may generate several voxels so we make a uniform choice, if necessary.

Example 5.6. APPLYING THE FOUR OPERATORS TO A 2–CORNER VOXEL.
To clarify this procedure, consider updating a voxel that currently contains two black corners, as shown on the left in Figure 5-9. The results of applying each of the operators is shown and each operator can generate several voxels, though only one is randomly chosen in practice.

Suppose the add-delete operator is chosen, for example. Applying the add operator means flipping the colour of the white corner with the greatest number of black neighbours. There are four white corners which each have one black neighbour and they are shown in the bottom row of Figure 5-9. Now apply the delete operator to each of these four 3–corner voxels. For any of these 3–corner voxels, applying the delete operator means flipping the colour of the black corner with the least number of black neighbours. There are two choices in each case. So applying the add-delete operator to a 2–corner voxel

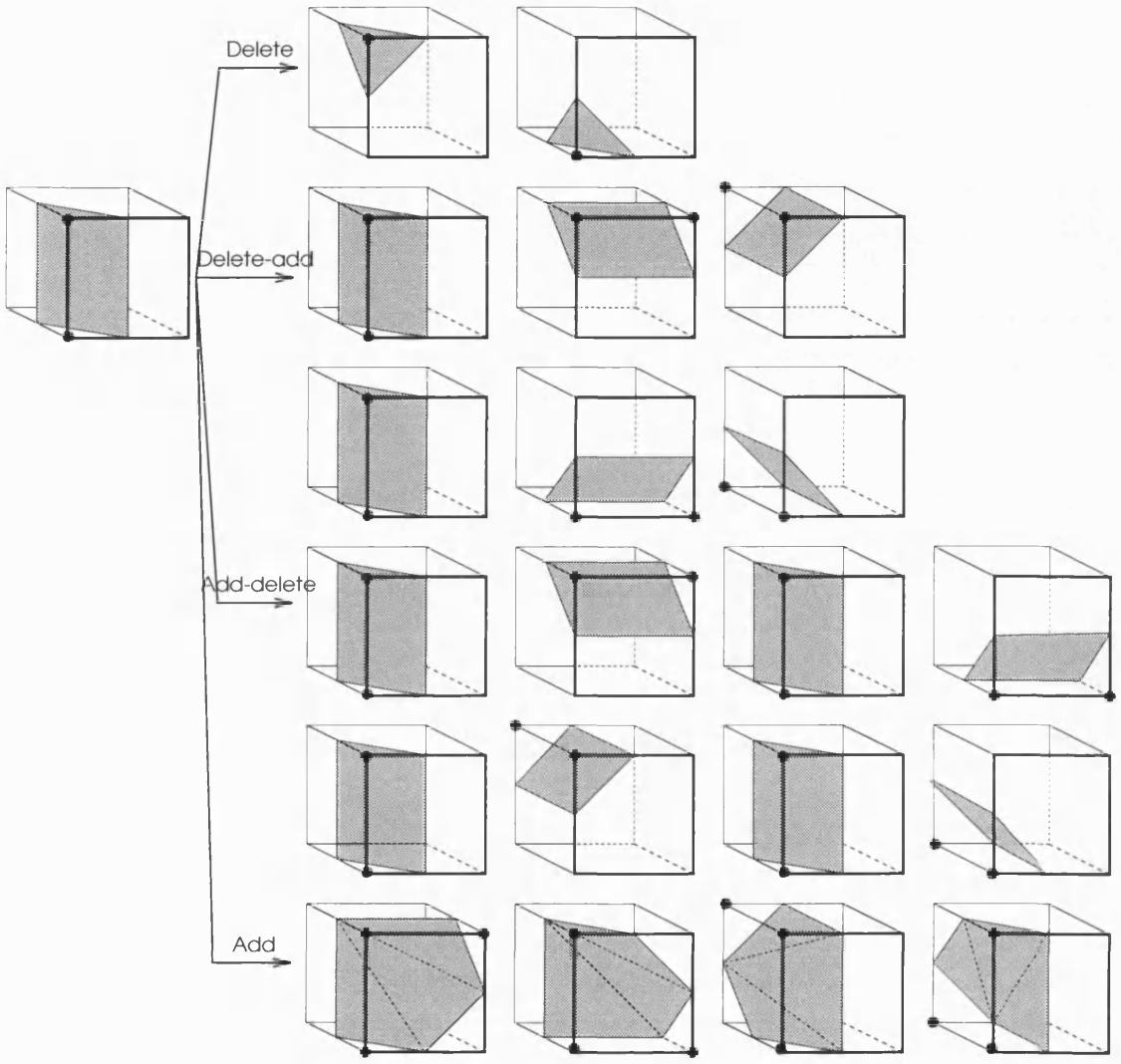

Figure 5-9: APPLYING THE FOUR OPERATORS TO A 2–CORNER VOXEL. Starting from the 2-corner voxel on the left, this figure shows all the voxels which can be reached in one move, by applying the add, delete, add-delete and delete-add operators. One of the four operators is randomly chosen and then one of the proposals that it generates is randomly chosen. The number and type of voxels generated varies, depending on the operator and number of black corners in the current voxel.

results in a total of eight possible updates for that voxel, shown in the second and third last rows of Figure 5-9. One of these proposals is then randomly proposed as the update for the current voxel.

Note that there are often several ways of generating a particular voxel, so the probability of proposing any one of the voxels shown in Figure 5-9 is not constant. For example, there are a total of twenty choices in Figure 5-9 but six are identical to the current state of the voxel. So the probability of a 2–corner voxel remaining unchanged is $\frac{5}{24} = \frac{1}{4}\frac{2}{6} + \frac{1}{4}\frac{4}{8}$. In contrast, there is only one occurrence of any of the 1–corner or 3–corner voxels, so the probability of any *one* of these voxels being proposed is $\frac{1}{16} = \frac{1}{4}\frac{1}{4}$. □

There is a now a two-tier procedure to generate a new proposal. First, randomly select one of four operators. Next, randomly select one of the proposals that that operator generates. Having generated the proposal, we then consider switching from our current voxel to the new voxel, depending on the energy of each proposal and the current temperature, as per usual.

Current number of corners	Probability of			
	add corner	delete corner	same number of corners	
			unchanged	changed
0	$\frac{1}{2}$	0	$\frac{1}{2}$	0
1	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{5}{32} = \frac{1}{4}\frac{1}{8} + \frac{1}{4}\frac{1}{2}$	$\frac{11}{32}$
2	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{5}{24} = \frac{1}{4}\frac{1}{3} + \frac{1}{4}\frac{1}{2}$	$\frac{7}{24}$
3	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{8} = \frac{1}{4}\frac{2}{8} + \frac{1}{4}\frac{1}{4}$	$\frac{3}{8}$
4	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	0
5	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{3}{8}$
6	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{5}{24}$	$\frac{7}{24}$
7	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{5}{32}$	$\frac{11}{32}$
8	0	$\frac{1}{2}$	$\frac{1}{2}$	0

Table 5.2: PROBABILITY OF MAKING A MOVE DURING STAGE 2. For an i–corner voxel, this table shows the probability of adding, deleting or keeping the same number of black corners, when generating a proposal. If the number of corners remains the same the probability of the new voxel being identical to the old voxel or not is also shown.

The list of all the voxels that can be generated from an i–corner voxel, where i varies from 0 to 8, is shown in Appendix A, for completeness. We summarise the results in Table 5.2. Given a voxel with i black corners, this table shows the probability of updating to a voxel with one more, one less or the same number of black corners. If the new voxel has the same number of black corners as the old voxel then they may or may not be identical and this table also shows the corresponding probabilities. For example, the most likely update for a 2–corner voxel is to remain a 2–corner voxel but with one of the two corners having moved. The probability of a voxel adding or deleting a black corner is $\frac{1}{4}$ because each operator is randomly selected. The probabilities for an i–corner voxel and an $(8-i)$–corner voxel are similar, for $i = 1, \ldots, 7$.

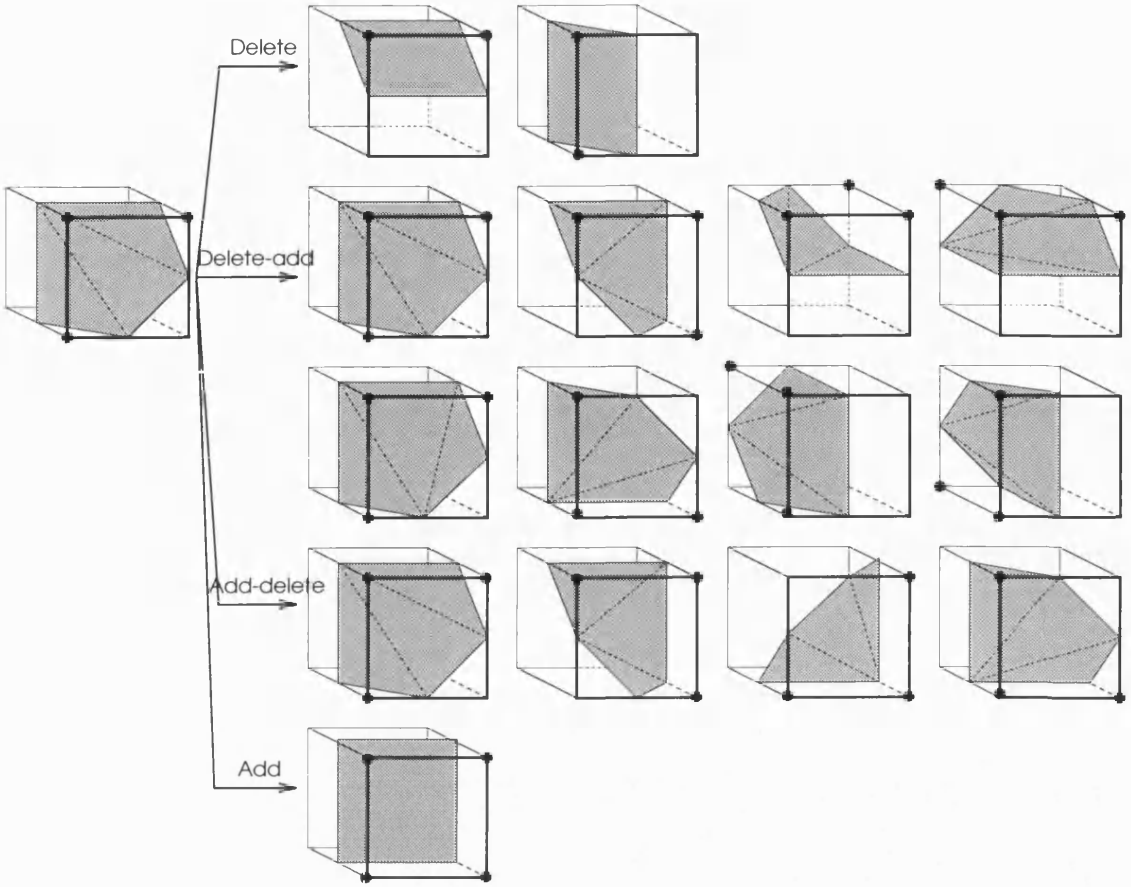

Figure 5-10: Applying the Four Operators to a 3–Corner Voxel. The layout of this figure is similar to that in Figure 5-9. However the probability of getting from the voxel on the left in Figure 5-9 to the voxel on the left in Figure 5-10 is *not* symmetric.

One side effect of this operator-based approach is that, unlike the 2D algorithm, Stage 2 does not consider all black or all white proposals unless the current state of that voxel has seven of the eight corners coloured black or white, respectively. For example, if the voxel is currently all white then it can remain unchanged or a single black corner can be added and there are eight corners from which to choose. Similarly, an all black voxel can remain unchanged or have a single black corner deleted. Note that the probability of updating from a 1–corner voxel to a 0–corner (all white) voxel is *not* the same as updating from a 0–corner voxel to a 1–corner voxel. Therefore, proposals are not always symmetric.

$$\Pr(\text{0–corner} \rightarrow \text{1–corner}) = 1/2, \qquad \text{see Figure A-1 on page 144,}$$
$$\text{and} \quad \Pr(\text{1–corner} \rightarrow \text{0–corner}) = 1/4, \qquad \text{see Figure A-2 on page 145.}$$

In addition, the operators may generate a different number of proposals, some of which may appear more than once. Consider moving from an image x whose i^{th} voxel is as shown on the left in Figure 5-9 to an image x' whose i^{th} voxel is as shown on the left in

Figure 5-10, where $x_{-i} = x'_{-i}$ and $x, x' \in \Omega_2$.

$$\Pr(x \to x') = \frac{1}{4}\frac{1}{4} \quad \text{and} \quad \Pr(x \to x') = \frac{1}{4}\frac{1}{2} \quad \Rightarrow \Pr(x \to x') \neq \Pr(x' \to x).$$

So the proposal matrix and hence the transition matrix are *not* symmetric. This excludes the Metropolis algorithm, so we consider the Hasting's algorithm instead (see §2.3.3 on page 27). To apply this algorithm, we use the four operators outlined above both to generate a proposal and to calculate the probability of selecting that proposal given the state of the current voxel. The acceptance probability in Equation (2.10b) requires the posterior energies for both proposals $f_{X|Y}(x, y)$ and $f_{X|Y}(x', y)$ from Equations (5.5).

Local minima problem in Stage 2

The operator based procedure for generating proposals did not work well in practice because the algorithm found it too easy to get caught in local minima. When it starts adding black to an all white voxel, it begins by changing just one corner to black and this corner is chosen at random. If a 'bad' choice is made then the algorithm finds it difficult to correct this 'mistake' later.

Example 5.7. LOCAL MINIMA PROBLEM IN STAGE 2.
Consider the following simple $2 \times 2 \times 3$ image, one half of which is black. The algorithm

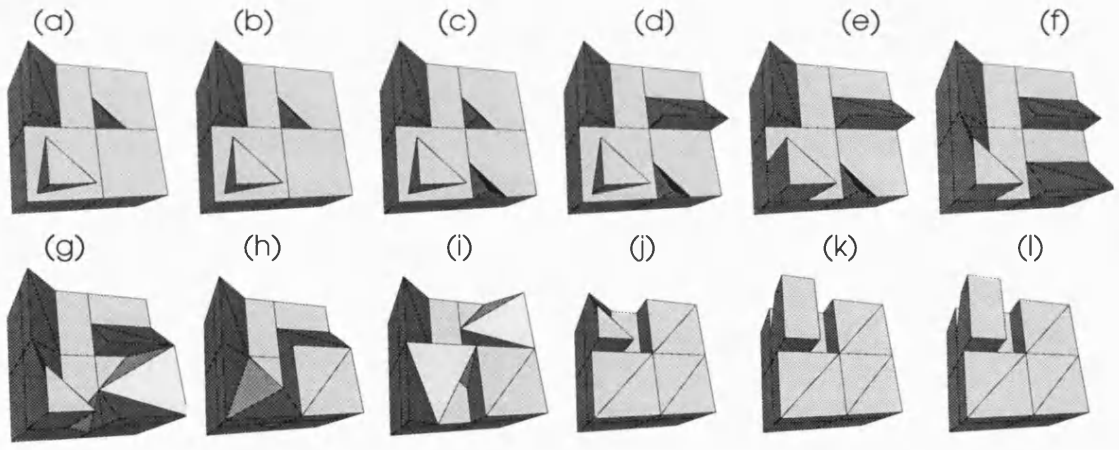

Figure 5-11: LOCAL MINIMA PROBLEM DURING STAGE 2. The true image is $2 \times 2 \times 3$ in size and is half black and half white. The algorithm is trying to place four half black voxels on top of four black voxels. The image is shown after each of the last 12 sweeps, using a temperature of zero, before it gets caught in a local minimum. After sweeps (a) to (i), the top, left voxel is a 2-corner voxel. During sweep (j), it adds an extra corner but in the wrong place. Undoing this 'bad' move turned out to be difficult for the algorithm, so it frequently fails to find the best solution to this simple problem.

encounters problems even with this simple problem. From Stage 1, four of the voxels are coloured black and eight white. The ideal Stage 2 solution is to leave the four black voxels

alone and place four half black voxels on top of them. However the algorithm can do this only by adding one black corner at a time. Figure 5-11 shows the image after each of the last 12 sweeps at a temperature of zero before the algorithm gets stuck in a local minimum. □

The problem seems to be that when the algorithm first starts adding black corners to an all white voxel, it can easily introduce the black corners in the wrong place and it has difficulty to get out of this position by undoing these moves later. On larger images this problem is even more pronounced, resulting in the main object in the reconstruction being surrounded by a proliferation of very small black objects. In theory, running the chain for longer should allow the algorithm to find its way out of these local minima; in practice, the large data sets and the amount of computation required for each update means that it takes too long to run hundreds of sweeps of the image.

Avoiding the local minima problem in Stage 2

To help overcome the problems of getting caught in local minima during Stage 2, we also consider an alternative strategy. It is a 'greedy' algorithm for optimisation, so it is not related to a MCMC sampler. Instead of selecting just one proposal, we simultaneously consider *all* the voxels that can be generated by applying the add and delete operators plus the possibility of remaining at the current voxel. The voxel which maximises $\phi(x)$ is always chosen as the new update. This requires a lot more work to update a single voxel but the algorithm has a much better chance of selecting the best corner to add or delete, thus avoiding very poor local minima. Given any i–corner voxel, the list of proposals considered when updating from it, are similar to those shown in Appendix A. The difference is that the add-delete and the delete-add operators are ignored, to reduce the length of the list of voxels that have to be evaluated.

Example 5.8. AVOIDING THE LOCAL MINIMA PROBLEM IN STAGE 2.
Figure 5-12 is similar to Figure 5-11 but each update considers all the voxels that can

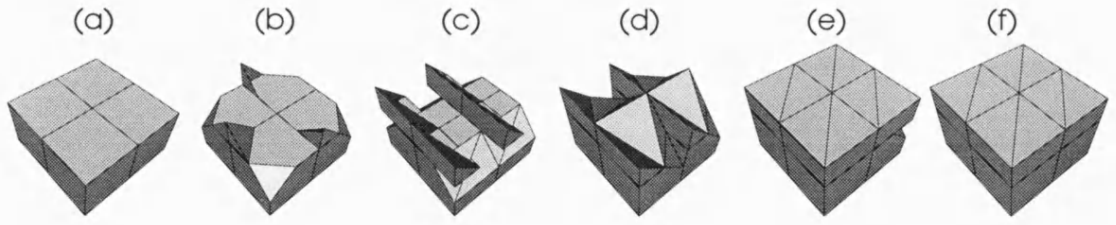

Figure 5-12: AVOIDING THE LOCAL MINIMA PROBLEM IN STAGE 2. This figure is similar to Figure 5-11 but each update considers *all* the voxels that can be generated by adding or deleting a black corner or keeping the current voxel. The algorithm successfully finds the solution to this simple problem each time.

be generated by adding or deleting a black corner or keeping the current voxel. At a

temperature of zero, the solution is found in 6 sweeps. Plot (a) shows the answer from Stage 1, Plots (b) to (f) show the next five sweeps from Stage 2. In Plots (b) and (c), the bottom four voxels switch from 8–corner voxels to 6–corner voxels as this will reduce the prior penalty. For the top four voxels, the number of black corners increases by one during each of the sweeps in Plots (b) to (e). This then encourages the bottom four voxels to switch back from 6–corner voxels to 8–corner voxels. □

This version of the Stage 2 algorithm successfully finds the solution to this simple problem each time. However, it should be noted that this is a strictly-uphill algorithm that converges on a local minimum.

Example 5.9. STAGE 2 RECONSTRUCTION OF A 3D OBJECT.
The Stage 2 reconstruction of the record in Figure 5-1 is shown in Figure 5-13. A smoothing

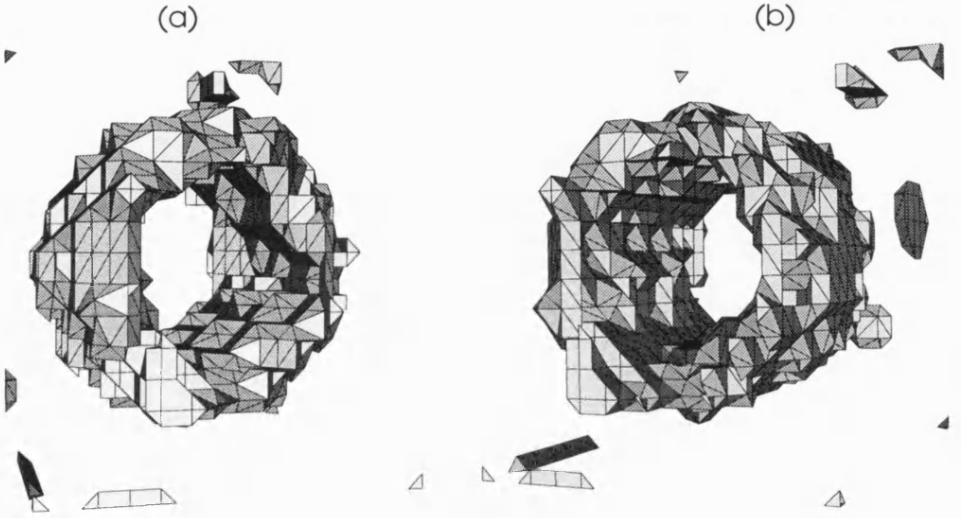

Figure 5-13: STAGE 2 RECONSTRUCTION OF A 3D OBJECT. For the data discussed in §5.1.1, this figure shows a reconstruction of the guard cell at the end of Stage 2. Again, the camera is shifted to the left in Plot (a) and to the right in Plot (b) to improve the depth information. This stage introduces surfaces through voxels but they do not necessarily link up.

parameter of $\beta = 100$ is used, just as in Figure 5-3.

Stage 1 has the effect of smoothing out the thresholded image in Figure 5-1. Stage 2 allows internal faces to be added to a voxel to split two regions of colour within that voxel. There appear to be several isolated objects just a few voxels in size. Some of these are located on the image boundary where the prior penalty is reduced because there are fewer neighbouring voxels. □

An alternative for selecting proposals during Stage 2

In theory, any mechanism that generates sensible proposals could be considered. In practice, the computational difficulties associated with programming in 3D make it very

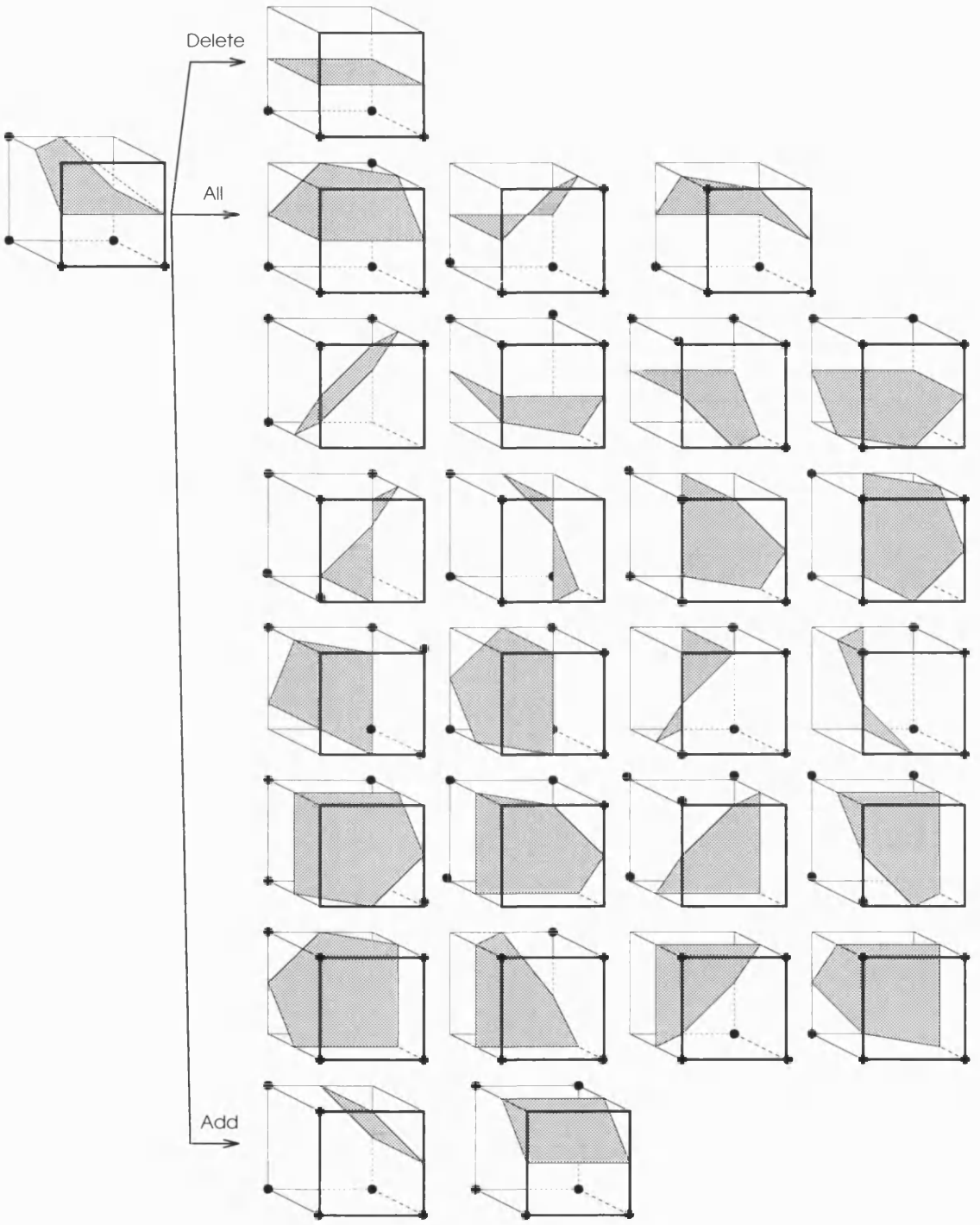

Figure 5-14: A 5-Corner Proposal Mechanism that was Rejected. This is an example of a 5–corner proposal mechanism that considers *all* 5–corner proposals (All) plus the possibility of adding or deleting a corner. Some proposals relocate the volume within the voxel, so they have a relatively large prior penalty and are unlikely to be accepted.

desirable that any potential proposal is likely to be accepted and can be generated easily. Suppose the volume in a 5–corner voxel is approximately correct, we must then decide the location for the volume within that voxel. A list of all possible 5–corner voxels would be large and some proposals would be unreasonable. Figure 5-14 shows the list of *all* 5–corner proposals. In addition, we need to consider changing the number of black corners, so the voxels that can be generated by applying the add and delete operators to the current 5–corner voxel are also shown. Some of these voxels are unlikely to be accepted because the location of the volume within the voxel does not agree with neighbouring voxels, so the prior penalty is relatively large. Also, generating this list may not be straightforward. Perhaps lists of proposals could be generated and stored from the outset but if separate code is needed for each i–corner voxel then the total amount of code could be substantial. Contrast the proposals generated by this more straightforward but heavy-handed approach with the proposals generated by the four operators shown in Figure A-6 on page 149. In the latter case, the list contains proposals each of which is likely to be accepted, if chosen; yet the list is shorter than that in Figure 5-14.

5.3.6 Stage 3: The conversion algorithm

Suppose an image in Ω_2 is also in the set Ω. Then at every point in the image where eight voxel corners meet, the same colour must be present in the eight corners. Similarly, where two or four voxel corners meet around the boundary of the image. It is straightforward to check that this is a sufficient condition for an image in Ω_2 to be in Ω and we base our conversion algorithm on this fact.

However, the conversion algorithm is less straightforward than in the 2D case because it is more difficult to link the internal faces in neighbouring voxels together.

Algorithm 5.2. CONVERSION FROM Ω_2 TO Ω.

1. *Use the final reconstruction from Stage 2 to assign a colour to each of the eight corners of each voxel.*

2. *Sweep the image by visiting each voxel corner in turn. Note the colours of the eight adjacent voxel corners (or zero, two or four adjacent corners around the boundary).*

   - *If one colour dominates by occupying more than four corners then re-colour the minority corners to agree with the majority.*
   - *Otherwise, the colours are evenly split. So calculate the average value of the records associated with all the voxels concerned and colour all corners with the colour that lies closest to the average record value.*

3. *Sweep the image. During the sweep, visit voxels in a* raster scan *taking slices from the bottom to the top of the image, then from the back to the front within each slice and then from the left to the right along each row.*

- *If the eight colours in the corners of a voxel correspond to a legal voxel colouring, assign this colouring to the voxel.*
- *Otherwise, choose the vertex that 'has the least effect' on previously processed voxels and colour it black. Figure 5-15 shows the vertex order used. After changing each corner, if the voxel is legal then goto the next voxel. If not change the next corner on the list.*

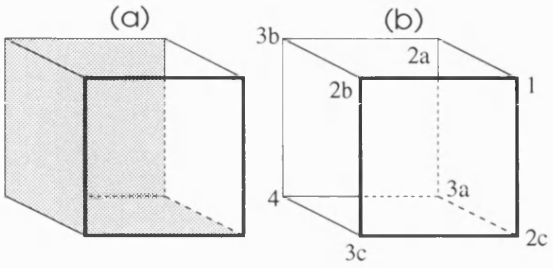

Figure 5-15: VERTEX ORDER FOR THE CONVERSION ALGORITHM. *For a voxel not lying near the image boundary, Plot (a) uses grey-shading to show the three neighbouring voxels that have already been updated. Plot (b) shows the order for updating the vertices that have the least effect on the updated neighbours. Corner 1 is the first to change to black, then the corners labelled 2a, 2b, 2c, 3a, 3b, 3c and 4.*

Remark. Note that the colouring of the current voxel eventually becomes legal because the voxel is then all black.

4. *Sweep the image repeating the previous step. Stop when the image is swept without changing a voxel.* □

The number of black corners increases monotonically during this process. It follows that the algorithm will converge because failure to converge at any intermediate point will result in an all black image which is a member of the image space Ω.

Step 2 is a relatively delicate way of resolving problem areas in the image. At the end of this step, we not be in Ω_2 or Ω, but we expect only a few illegal voxels. Steps 3 and 4 are crude in comparison to Step 2 but they are an effective way to produce an image in Ω. The algorithm may benefit from further refinement.

5.3.7 Stage 4: Subvoxel adjustments

At the end of Stage 3, the boundary around each object in an image is identified by a linked, sequence of triangles, each of whose vertices lies midway along a voxel edge. Each voxel is either all black, all white or has a single surface through it which joins with the surfaces passing through its neighbours.

Example 5.10. SOME EXAMPLES OF STAGE 4 VOXELS.
In Stage 2, vertices on the inner faces are constrained to lie midway along the voxel boundary. In Stage 4, this restriction is removed. Figure 5-16 shows some examples of Stage 4 voxels. Plots (a) to (g) have one to seven black corners, respectively.

The black part of some voxels is convex in shape, such as Plot (a). For the voxels shown in Plots (b), (d), (f) and (g), the shape of the black region is star-shaped from any

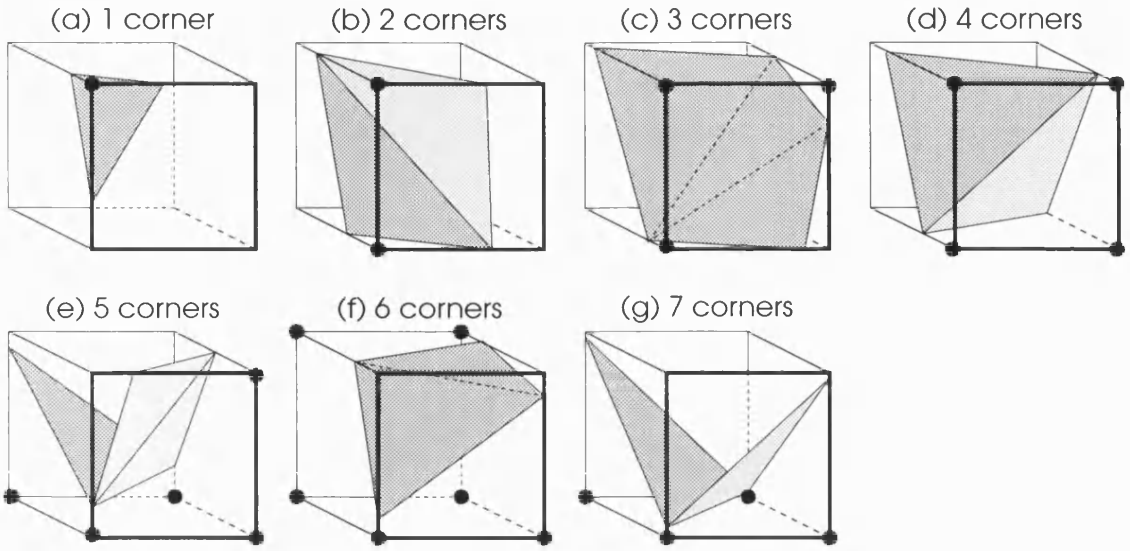

Figure 5-16: SOME EXAMPLES OF STAGE 4 VOXELS. This figure shows some examples of Stage 4 voxels with one to seven black corners. The vertices of the internal faces are *not* constrained to lie midway along an edge between two black corners. The black region in each voxel is star-shaped from some point within that part of the voxel. The voxel apex is a suitable point in every case.

point along the common edge between the two internal faces. Even more awkwardly, the 3–corner and 5–corner voxels in Plots (c) and (e) are star-shaped only from the common vertex shared by the three internal faces. In all cases, the region of black in a voxel is star shaped from the apex of the voxel, as defined in §5.3.4 on page 113. □

Each external face on a voxel exactly matches the opposing external face in its neighbouring voxel, so the prior penalty for any voxel arises from the area of the triangles on its inner faces, if any.

In this final stage, the vertices that define the inner faces in a voxel are free to move along the voxel boundary. We do not allow vertices to move into neighbouring voxels. The location of the vertices is decided in a manner similar to that used in the final stage of the 2D algorithm. Thus, the image space is effectively reduced to a subset of Ω by the beginning of Stage 4. This reflects the work done during the earlier stages. Consequently, the starting temperature for a simulated annealing search can be set to a high value in order to explore this subspace of Ω fully. Details of this application of simulated annealing are described in §2.3.3.

The simulated annealing algorithm is based on a MCMC sampling algorithm. We use the Metropolis algorithm here. In making a proposal for a move, a 3D point along the voxel edge upon which the vertex initially lies is chosen at random. The change in posterior probability decides the probability of moving the vertex to that position. A sweep of the image means updating every vertex location once. Typically the number of updates required to sweep an image is far less than the number of voxels in the image, as

only a few voxels contain two colours. This reduction in computation is more pronounced in 3D than in 2D and is especially important given the larger datasets in 3D.

The algorithm for updating an image in Ω is the 3D analogy of Algorithm 4.3 on page 79. The details for this 3D algorithm are omitted because they are rather similar to the 2D algorithm. The primary difference from the 2D algorithm is that *four* voxels need to be updated to change a single vertex.

Example 5.11. A STAGE 4 MOVE.
The effect of changing a single vertex is illustrated in Figure 5-17. The colouring of

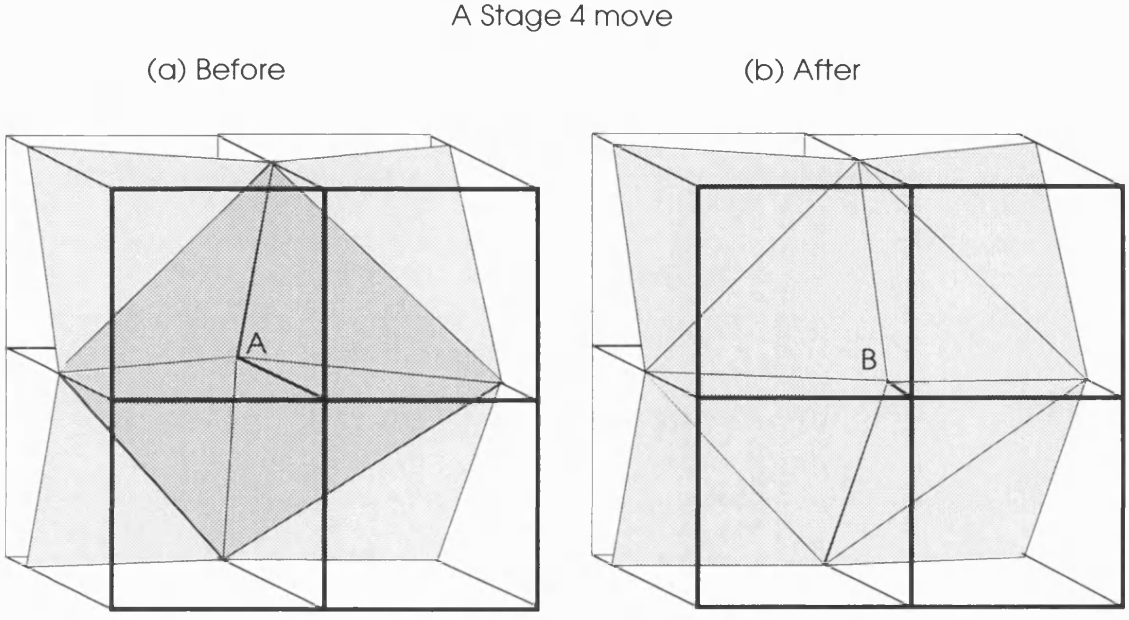

Figure 5-17: A STAGE 4 MOVE. This figure shows a typical 'before' and 'after' Metropolis move for Stage 4. The internal faces in each voxel link with internal faces in neighbouring voxels so the prior for each voxel associated with the external faces is zero. Only one vertex is moved during any update, from A to B, but four voxels need to be considered simultaneously.

four voxels is affected as the two edge-segments meeting at the vertex in question are repositioned from A to B, where B is chosen randomly along the voxel edge. □

Calculating the volume of a voxel in Ω

The shape that voxels are allowed to take during Stage 4 affects the choice of algorithm used to calculate the volume occupied by the two regions of different colour inside a voxel. The volume of black in any voxel in Ω can be partitioned using a set of tetrahedrons. In every case, the apex of the voxel is a suitable point from which to partition the voxel into tetrahedrons because the voxel apex can also be defined to be the apex for each of the tetrahedrons. The base of each tetrahedron is formed from three linked vertices lying on an external face of the voxel. Calculating and accumulating the volume for each tetrahedron gives the volume of the black region in *any* legal voxel.

Algorithm 5.3. VOLUME OF BLACK IN A VOXEL.

- *Given any voxel,*
  - *If it is all white then the volume of black in the voxel is zero. Exit.*
  - *Otherwise decide which vertex is the apex and label it* p_a.
- *For each external face in the voxel that does* not *contain the apex:*
  - *Use three successive vertices on that face,* p_1, p_2 *and* p_3, *as the base of a tetrahedron.*
  - *Calculate the three vectors from the apex to each of the three vertices on the external face,* $\mathbf{v_1} = p_1 - p_a$, $\mathbf{v_2} = p_2 - p_a$ *and* $\mathbf{v_3} = p_3 - p_a$.
  - *The volume for that tetrahedron is* $\frac{1}{6}|\mathbf{v_1} \cdot (\mathbf{v_2} \otimes \mathbf{v_3})|$, *where '* $\cdot$ *' is the dot product and '* $\otimes$ *' is the cross product of two vectors.*
  - *Repeat for any triplet of successive vertices remaining on that external face and accumulate the answer.*
- *Repeat for any other external face in the voxel that does not contain the apex and accumulate the answer. Exit.* □

For Stages 1 or 2, this algorithm partitions the black region in *any* legal voxel into tetrahedrons. The volume of white in any voxel is the volume of the entire voxel minus the volume of the black region inside the voxel. The volume for the entire voxel depends on the relative sizes of one unit in the x, y and z directions. In turn, this is dependent on the recording sensor. Usually the x and y scales are equal but the z scale is larger.

Example 5.12. A STAGE 4 RECONSTRUCTION.
Figure 5-18 shows a reconstruction from each stage of Algorithm 5.1, for a simple $5 \times 5 \times 5$ record. The record is an approximation to a sphere just over one voxel in size, to which some noise has been added $\sigma^2 = 1$. (The small size of the record makes it possible to see greater detail in the reconstruction.) Plot (a) shows a single voxel which lies inside the sphere. No other voxel has a sufficiently large region inside the sphere to justify colouring it as an all black voxel. During Stage 2, some subvoxel detail is added to the reconstruction. The reconstruction is viewed from two different angles in Plots (b) and (c). In this reconstruction, Stage 2 of the algorithm has failed to link all the internal surfaces. Stage 3 forces the surfaces to link together to form a regular shaped object, so only one view is shown (Plot (d)). Two views of the Stage 4 reconstruction are shown in Plots (e) and (f). The visible irregularities of the Stage 4 surfaces are partly due to the noise in the record and partly due to the temperature not being close to zero. The irregular shape of the reconstruction gradually disappears with decreasing temperature. At a temperature of zero, an approximately regular shaped object is formed to minimise the surface area. This reconstruction looks like Plot (d), so it is not shown. □

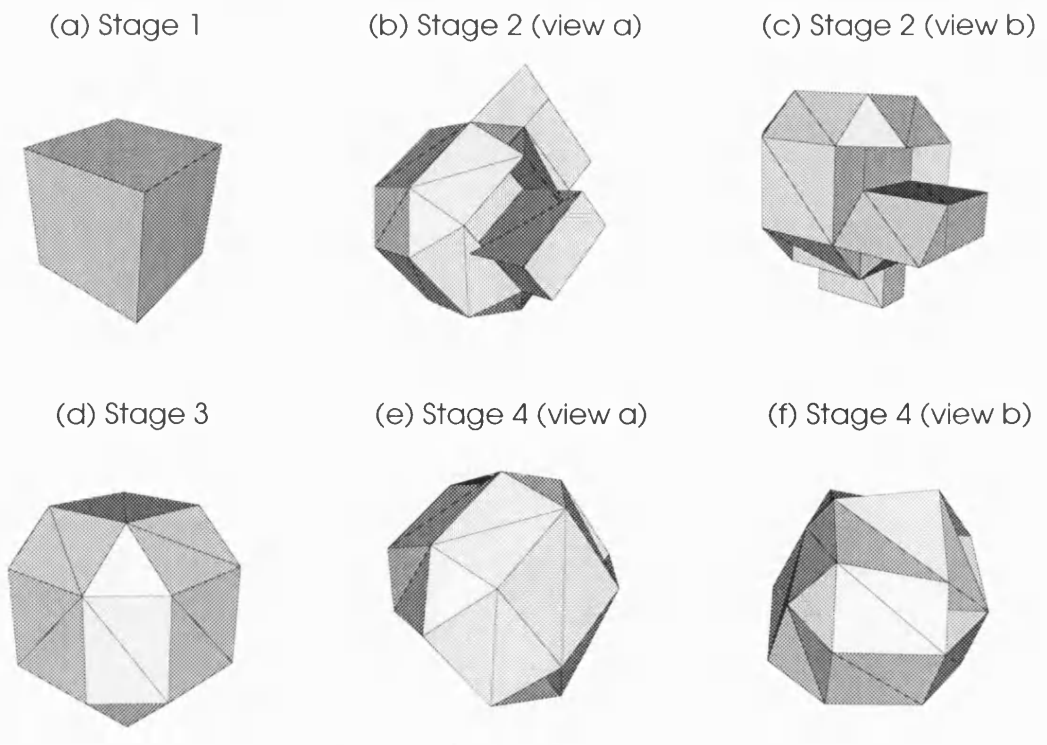

Figure 5-18: A STAGE 4 RECONSTRUCTION. This figure shows each of the four stages from Algorithm 5.1. To illustrate the detail, a $5 \times 5 \times 5$ record is used to reconstruct a spherical object. Stage 1 produces a single voxel lying in the center of the sphere. Stages 2 and 3 build regions of black into the voxels surrounding the centre voxel. Stage 4 allows the vertices to move, producing a less regular object.

This example demonstrates that Algorithm 5.1 is effective in principle. The guard cell example, from §5.1.1, could be put through Stages 3 and 4. This work has not yet been done.

5.4 Final remarks

5.4.1 Contrasting our 3D subvoxel model with other 3D methods

The primary difference between our method and the other previously proposed methods is that ours is an explicitly subvoxel model. The other methods only model the data at the full voxel level or, in some cases, the data are aggregated and analysed on a coarser level than the underlying record. In addition, we model the noise in the image explicitly.

The model and algorithm are fully three dimensional and automatic, once the parameters have been estimated, such as the noise variance. This contrasts with the manual approach in §5.1.2 and it avoids the complications associated with contour stitching because we do not view the data as simply a sequence of 2D slices.

The marching cubes algorithm has no underlying model, it simply thresholds and

triangulates the data. Our method is not an interpolation technique, because there are explicit prior and degradation models. The definition for a voxel in the marching cube algorithm differs from that used in our method. The marching-cubes algorithm uses eight record elements to form the vertices of a voxel; our algorithm forms a voxel around each record element. Our model is not necessarily deterministic, unlike marching-cubes. Also in contrast to marching-cubes, all possible triangulations are not considered. The restriction is that at most one surface is allowed within a voxel, analogous to at most a single line across a pixel in 2D.

Our model uses prior information, just like Kass, Witkin, and Terzopoulos's 'snakes' model. Unlike splines, the internal energy is not a global operation so there are no associated computational difficulties. If global optimisation is used, the algorithm is less sensitive to its starting point but extra computational costs are incurred for global solutions.

There are several features in our model that contrast with the 'deforming primitive templates' model. Our model does not insist on a single, closed surface at all stages and there is no template that has be positioned within the volume. In addition, our model can deal with highly non-convex surfaces. The amount of computation our algorithm needs depends primarily on the size of the original data not the size, orientation nor complexity of the object, subject to subvoxel limitations. However, our algorithm does requires more computation and different levels of resolution are not possible.

The number of triangles in the final reconstruction is proportional to the size of the object and inversely proportional to the size of a voxel. Also, the number of triangles does not depend on the complexity or orientation of the object, unlike most of the other 3D models. Reconstructions are not self-intersecting because a voxel is never allowed to have more than one surface through it. Our model can deal with any number of objects, even one object inside another, subject to the subvoxel assumption of at most one surface through a voxel. We restrict ourselves to objects that are small relative to the sensor resolution and expect advances in computing speed and memory to speed up the algorithm over time.

5.4.2 Outstanding issues

We state some of the problems that are specifically related to this chapter. Some of these issues are discussed in greater detail in the next chapter.

The algorithm Overall, we make no assumptions about the shape, orientation or number of objects in a 3D data set and get a fully 3D reconstruction after Stages 1 and 2.

In theory, the initial algorithm for Stage 2 should produce reconstructions that are the 3D analogy of the results in the previous chapter. In practice, the algorithm took a long time to run. There are at least three reasons for the slow speed at which the sampler works:

- The number of choices for updating each voxel is very large so it takes time to sample from this set.

- The amount of computation needed to choose an update for a voxel is much greater than in the 2D algorithm. This problem could be alleviated by using lookup tables.
- The data sets are much larger than in the 2D algorithm.

Nevertheless, the basic framework is in place: we define the statistical model and respect it during each stage of the algorithm, while optimising over the underlying distribution. We use an alternative strictly downhill algorithm (see §5.3.5 on page 123) to speed up Stage 2. This algorithm looks for the best change within a given set at each voxel update.

Basic software tools An essential prerequisite for 3D modelling of any kind is software that offers primitive, 3D objects, which can be used as basic building blocks, such as drawing axes or writing labels. Without this support, it is necessary to build even the most elementary objects, such as 3D points or polygons before any real image processing can begin. This constraint lengthened the project outlined in this chapter by an order of magnitude.

Sensor characteristics The non-isotropic resolution characteristics of most optical microscope data become obvious at viewing angles that are perpendicular to the imaging data. In practice this means that the lower resolution along the optical axis stretches the flexibility of most algorithms, including the algorithm outlined in this chapter.

Multiple intensity values Another drawback with all these models is that every record value is classified as either inside or outside the reconstruction. It may be that the surface cannot be modelled accurately using single intensity values, as often happens in 3D biological structures.

Approximating data Most data sets are large, so an algorithm that requires less computation is an advantage but this requirement will become less important as processing power increases. The algorithm in this chapter is restricted to generating reconstructions where each element is at most the size of a voxel. Therefore, it does not approximate the data. It is only suitable for rendering structures that are small relative to the size of a voxel but this is usually the case with any subvoxel algorithm.

Data measurement The algorithm in this chapter is not affected by the complexity of the underlying 3D structure, at least down to a subvoxel level. This contrasts with most other 3D algorithms. Once generated, a reconstruction from this algorithm can be used as the starting point for subsequent quantitative analysis such as rapid visualisation and rotation, shape recognition, subjected to a series of geometric operations or geometric measurements such as volume, length and integrated intensity measurements. It might also be interesting to compare our final reconstruction of an object to a 3D template of the object.

Convergence and efficiency of MCMC There are convergence issues that need to be considered with 3D MCMC. The large dimensionality of images may make monitoring of convergence difficult. The speed of convergence seems to decrease as the dimensionality increases (Green 1995a), so this may be a problem in three dimensions (see §2.3.7 on page 33).

5.5 Summary of chapter

- The 3D algorithm is outlined very briefly and an example is given of a 3D object.
- Existing 3D methods are outlined.
- An image space along with the prior, likelihood and posterior probabilities are defined.
- The 3D algorithm is defined in detail.
  - An example is given of a Stage 1 reconstruction.
  - Examples are given of images from Ω_2 and the different types of voxels that can arise are classified.
  - Two basic operators are defined. They are applied to the current voxel to generate a proposal for updating that voxel. This allows the image space to be explored during Stage 2 by changing just one corner of a voxel at time without requiring a complex algorithm for dealing with all the special cases that arise. Examples are given of the sort of proposals that the operators can generate.
  - The initial algorithm is not very successful because it can get trapped in a local minimum too easily. An alternative algorithm that considers several potential updates simultaneously is more successful.
  - An algorithm to convert an image from Ω_2 into the image space Ω is outlined.
  - An example is given of a typical updating move during Stage 4, along with some examples of Stage 4 voxels. The algorithm used to calculate the volume of the black or white region in a voxel is stated.
  - A reconstruction of a small object illustrates the movement of vertices at the subvoxel level.
- The issues raised in this chapter, are discussed.
  - The model and algorithm is contrasted with the other models outlined at the start of the chapter. Note that these other models do not attempt to explicitly model the data to a subvoxel level.
  - Outstanding issues are listed and briefly discussed but see Chapter 6 for more detail.

Chapter 6

Conclusions and further work

We shall not cease from exploration
And the end of all our exploring
Will be to arrive where we started
And know the place for the first time.
Thomas Stearns Eliot

I never see what has been done;
I only see what remains to be done.
Madame Curie

Chapter 1

Conclusions

- Even from the brief review of image analysis in Chapter 1, it is clear that the visualisation and subsequent analysis of physical processes by recording an image of the data is a technique of growing importance to the scientific community. Consequently, mathematical models that allow this process to become automated or simply accelerated will continue to be in great demand.

- Although this thesis has a narrow focus within the field of image analysis, subpixel edge detection is still a technique with numerous potential applications in widely varying scientific fields. The methods presented in this thesis are quite different from those cited in §1.3.3 and §5.1.2, so they provide an alternative approach to subpixel reconstruction.

- The objectives in §1.4.1 have generally been met. Various models and algorithms have been proposed and demonstrated for building edges onto 2D images and surfaces through 3D volumes that allow pixels or voxels to be split into two regions. The models are strictly local in nature making them easier to implement and more robust, especially with respect to the number and shape of the objects in the image. Various parameters can be altered in the models, such as using a strictly downhill algorithm to give fast approximate answers. Ideally, an automatic procedure for dealing with all of the parameters is needed, as is a detailed sensitivity analysis of the parameters. These are areas for future work. Several different techniques are employed to visualise the data. In 2D, these methods are fast, given the volume of data in the record, but

the data volume is more of a constraint in 3D.

Chapter 2

Conclusions

- The richness of the Bayesian/Markov chain approach to image analysis is amply demonstrated by its flexibility in the different algorithms in Chapters 3, 4 and 5 of this thesis.

Further work

- Alternative priors could be considered. For the continuous, 2D subpixel model in Chapter 2, minimising the length of the edge around an object could be replaced by a prior that penalises divergence in the angle between two linked edges in neighbouring pixels from 180°.

- A different estimator to the MAP estimator could be considered. In particular, establishing credibility regions for the edge around an object would be a major step forward. The posterior distribution could be sampled rather than simply optimised to produce estimated confidence intervals for the reconstructions. This would require the reconstructions from the Markov chain to be stored after an initial burning-in period.

Chapter 3

Conclusions

- The discrete, 2D subpixel model is more intuitive than the continuous model in Chapter 4. However, extending the discrete, 2D subpixel model to three dimensions is not a realistic possibility due to the computational demands of the model.

- Ensuring consistency between different levels in the cascade is worthwhile because it allows comparisons to be made between all the reconstructions. This allows the benefit of the subpixel reconstruction to be objectively measured, as the difference between the first and final level reconstructions.

- In comparison to the continuous, 2D subpixel model, the discrete model requires more memory to store and takes longer to visualise the reconstructions, as the algorithm proceeds to the finest level of detail.

Further work

- To make this model applicable to real data, it is essential to cater for blurring.

- The discrete and continuous, 2D models could be compared to see which is faster, more accurate, *etc.*

Chapter 4

Conclusions

- This algorithm condenses the data in the record while still retaining most of the information in the record. This makes subsequent analysis and visualisation much faster. In this respect, it outperforms the algorithm in Chapter 3.

- The examples demonstrate that the algorithm is robust to the number and shape of the objects in the record.

Further work

- The effective amount of resolution at the subpixel scale is limited by the assumption of piecewise linearity across pixels. Further work is needed to decide if this assumption is always justified. It is likely to rely on the particular application.

- Stage 3 is awkward and perhaps unnecessary. One way of ensuring that the reconstruction from Stage 2 is in the image space for Stage 4 is to allow up to two lines across a pixel. Whether the extra work that this entails is justified is an open question.

Chapter 5

Conclusions

- To explore different models and algorithms in a reasonable time-frame, a library of basic 3D image manipulation procedures is essential. Commercial organisations do provide such software but costs are prohibitive. Until cheap platforms are available, 3D data manipulation will continue to be laborious. This point is critical.

- There are problems with visualising 3D data. It is necessary to draw the geometric features of the black region in *every* voxel, even those obscured by occlusion. This is inefficient. Alternative ways of visualising 3D data need to be investigated.

Further work

- Future efforts should concentrate on model quality and how to measure it (e.g. computing time, computing memory, number of edges/triangles needed to cover a sample object) and alternative datasets. For example, would the algorithm be much more successful if we used non-blurred or low noise images? Alternative datasets

which are essentially binary in nature would be particularly helpful. This may mean looking at non-biological data sets.

- The volume of the data sets in 3D is a serious obstacle to practical 3D imaging. Algorithms that aggregate or approximate the data are likely to be more appealing to practitioners, until computer processing power improves.

- The sensitivity of the model to the assumptions made about the data should be considered further and the model should be robust with respect to extreme behaviour. In particular, it seems necessary to allow for non-stationarity in the blurring. This is because blurring along the z-axis is often different to the blurring kernel on the x and y axes.

- The subvoxel reconstruction requires a large number of triangles to specify an image in Ω. Can the number of triangles be reduced without compromising the level of detail in the reconstruction?

- There are convergence issues that need to be considered with 3D MCMC. The large dimensionality of images may make monitoring of convergence difficult (see §2.3.7 on page 33). Perhaps many more sweeps are needed compared to two dimensional problems? If so, the greater volume of data in 3D will compound this problem.

- How can time varying models be produced (e.g. to model the guard cell dataset as the cell closes)?

Some final comments

- If the objective is only to minimise the energy then subpixel models do much better than full pixel models because they can simultaneously reduce both the prior and the likelihood in situations where a full pixel model would have to balance the two penalties.

- Once generated, a reconstruction can be used as the starting point for subsequent quantitative analysis such as visualisation, shape recognition, subjected to a series of geometric operations or geometric measurements such as volume, length and integrated intensity measurements.

- The most important area for future development is the evaluation of the models and algorithms in a quantitative, systematic and extensive way. This includes comparing the results in this thesis with other competing models. Standard test cases could be analysed using a variety of subpixel models and the experimental results contrasted. For example, it might be concluded that a model is accurate to $\frac{1}{10}^{\text{th}}$ of a pixel, say. In practice this would not be easy. Great care is needed to compare the models presented in this thesis with other subpixel models on a consistent basis. Also, the software to implement the models proposed by others is often not readily available.

Due to the complexity of many of the algorithms that have been proposed (see §1.3.3 and §5.1.2), it is not practical to rewrite the corresponding software.

The amount of CPU time consumed by the algorithms is proportional to the number of simulated annealing sweeps. This user-selected parameter could be investigated to speed up simulations. In particular, the run times relative to other algorithms need to be tabulated because the algorithms presented in this thesis are computationally complex.

The current implementation is designed so that various ideas and geometrical relationships can be tested with relative ease. Because of this the current implementation of the various algorithms is far from optimal. Alternative data structures (see §B.1) could reduce the run-times by an order of magnitude. However, it is important not to become obsessed with computational speed. These models typically take several years to mature and in the meantime the speed of the underlying hardware will probably increase greatly.

- The methods work best for data where the total number of pixels or voxels is small compared to the total size of the image. Boundary representation has the advantage of a significant reduction in space requirements for the data.

- Not much is gained by developing fast algorithms for processing the data if the visualisation of the reconstruction is slow.

  The major problem in storing and manipulating 3D images is their size. A single section, of size 512×512, requires a quarter of a megabyte, assuming one byte per pixel. Often 3D image sections are of size 1024×1024 and there may be 50 sections. So the whole image is 10–150mb in size, ignoring compression techniques.

- The models presented in the thesis are not confined to a particular field of study. Analysing analogous subpixel problems in other fields would help greatly to improve and refine the models.

- Extending all the models to more than two colours is important but not necessarily essential. Morphology is a subject that relied on images being binary in nature (Serra 1982). More recently, this theory has been extended to deal with grey-level images. The approach is to order the colours, from 1 to n say and then deal with pairs of successive colours at a time, $\{(1,2),(2,3),\ldots,(n-1,n)\}$. For each pair of colours $(i, i+1)$, every pixel in the ranges $(1, i+0.5)$ and $(i+0.5, n)$ will tend to be classified as colours i and $i+1$, respectively. After all pairs have been processed, the different reconstructions can be 'laid on top of one another' to form a grey scale reconstruction. Perhaps a similar approach could be used with the subpixel models described here.

  Alternatively, an extra stage could be added to the model, where the choice of the two most likely colours is dependent on the colouring of the neighbourhood around a

pixel. However, there are some extra complications that need to be addressed, such as the possibility of three regions of different colours meeting inside a pixel.

Even then, there are many problems where the colours may blend continuously into each other in some parts of the image. The models in this thesis cannot deal with such cases.

- All the models need an objective method to choose the smoothing parameter β.
- It would be interesting to consider sampling rather than simply looking for the MAP estimate by optimising during Stage 4. This would be a distinct advantage over other subpixel algorithms by offering the possibility of confidence limits for the surface area and volume measurements.
- Another possibility to consider is feeding the answer from Stage 3 back into Stage 2 to see if the changes make during Stage 3 have any effect on the Stage 2 reconstruction.

Acknowledgements

If you wish in this world to advance,
Your merits you're bound to enhance,
You must stir it and stump it,
And blow your own trumpet
Or, trust me, you haven't a chance!
William Schwenck Gilbert

It is impossible to begin to learn that
which one thinks one already knows.
Epictetus

To Professor Christopher Jennison, for his guidance and encouragement while supervising this thesis.

To my colleagues and friends in the Statistics Group at Bath, for providing such a pleasant working environment. (Special thanks to Jeff for letting me practice on his thesis and for successfully culling the proliferating population of commas in this thesis!)

To my parents (Martin and Margaret), family and friends, for their support and encouragement.

To the School of Mathematics, for providing the infrastructure which is essential for this kind of research, especially the computing facilities.

To David Hitchcock at the Scottish Agricultural Statistical Services and Karl Ritz at the Scottish Crop Research Institute, for kindly supplying the microscopic fungus data analysed in Chapter 4.

To Mike Fricker and Nick White in the Department of Plant Sciences at Oxford University, for supplying the three-dimensional dataset of a plant guard cell that is discussed in Chapter 5.

To the Engineering and Physical Sciences Research Council (EPSRC) (formerly the Science and Engineering Research Council) and to the Graphical and Computational Statistics Section of the American Statistical Association for their financial support.

To the numerous, steep hills around the city of Bath, for the invigorating challenge that they provide for joggers.

To my amazement, for finally finishing this project and bringing this chapter of my life to an end.

Colophon

The typesetting relies on the $\mathcal{AMS}$-LaTeX macro package (AMS 1995) for LaTeX (Lamport 1986). The generic text font is 11pt Computer Modern Roman but other fonts from the same font family are used where required. For example, caption titles are set in small caps while the caption body is sans serif. The bibliography follows the Chicago citation style. An index is provided for cross-referencing.

A wide variety of languages are used to write the software used in this thesis:

- `C` (GNU V2.5.8) is the low level language used for the discrete, subpixel algorithms in Chapter 3 and some of Chapter 4.
- C++ (GNU V6.3.2) is the low level language used to for the algorithms in some of Chapter 4 and all of Chapter 5. A C++ library for combinatorial simulation called `LEDA` (V3.1.2) (Naeher 1995) proved to be invaluable (see §B.1 on page 153).
- The 2D images were visualised using mainly `SPlus` but the 'hand-drawn' figures are from `Xfig`.
- The only excursions out of SPlus for displaying data are the 3D images generated by `Geomview` , the well designed software of Stuart Levy, Tamara Munzner and Mark Phillips for direct manipulation of 3D graphics.
- Occasionally, results are derived from `Matlab` and `Maple`.
- All figures are stored in PostScript.
- The simulations were mostly run on a SUN Sparcserver 1000.

Appendix A

A complete list of Stage 2 proposals

To find out what happens to a system
when you interfere with it
you have to interfere with it
(not just passively observe it)
George E. P. Box

Investigators seem to have settled
for what is measurable instead of
what they would really like to know.
Edmund D. Pellegrino

For reference, this appendix contains the complete list of updates that are possible during Stage 2 of Algorithm 5.1 (see page 108 and §5.3.5 on page 117).

The set of potential updates depends on the number of black corners in the current voxel. This can vary from zero for an all white voxel to eight for an all black voxel. For convenience, each set of updates is visualised in a series of figures rather than being tabulated. The current state of the voxel is shown on the left of each figure. Each arrow emanating from the current voxel denotes one of the four operators discussed in §5.3.5. One operator is chosen at random. Each operator usually results in several potential choices, one of which is randomly chosen. The possibility of keeping the number of black corners the same is also shown. This achieved by applying either the add-delete of the delete-add operator.

Note, that the current voxel could be rotated to another position and this requires all the potential proposals on the right-hand side of each figure to be rotated in a similar manner. Consequently, the algorithm has to deal with many more possibilities than are shown here. To keep the number potential proposals manageable, it essential that any updating procedure is invariant to rotation. Such is the case with the algorithm in Chapter 5.

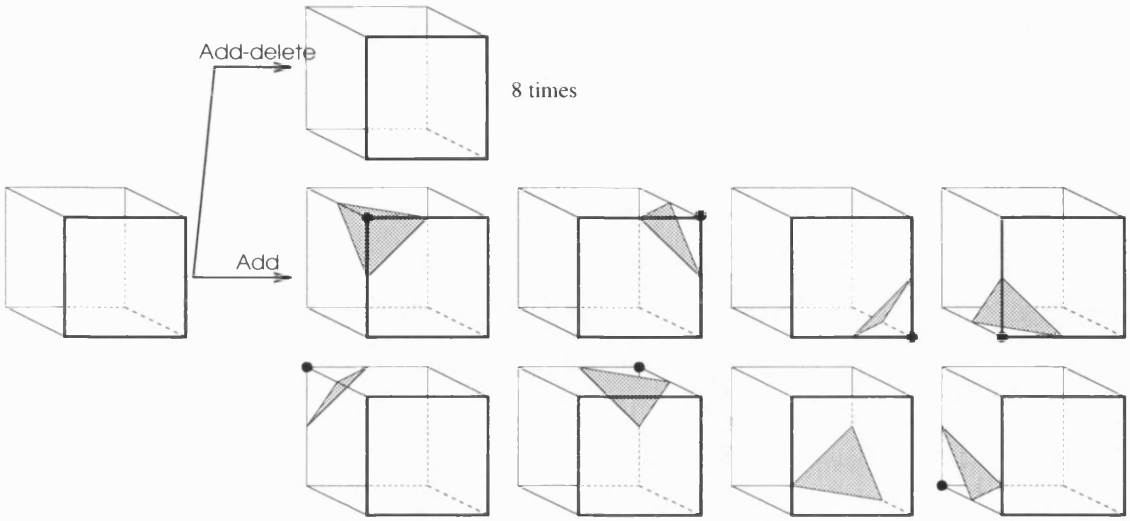

Figure A-1: PROPOSALS WHEN MOVING FROM A 0–CORNER VOXEL. This figure shows the list of updates for a particular *zero*-corner voxel, draw on the left. The arrows emanating from this voxel point to the list of potential updates formed by adding or adding and then deleting a black corner to the current voxel.

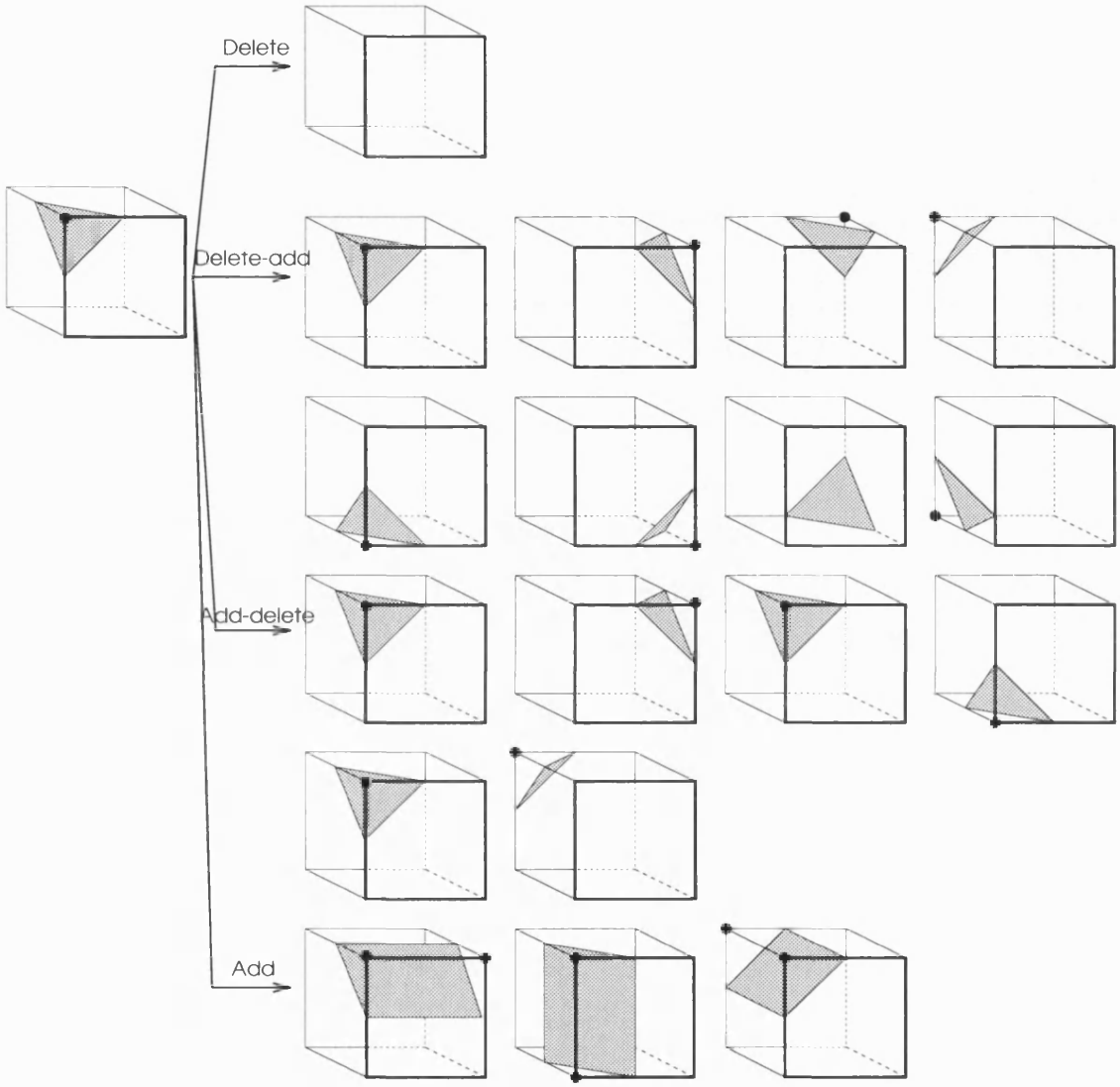

Figure A-2: PROPOSALS WHEN MOVING FROM A 1–CORNER VOXEL. This figure shows the list of updates for a particular *one*-corner voxel, draw on the left. The arrows emanating from this voxel point to point to the list of potential updates formed by deleting a black corner, keeping the same number of black corners (by deleting-adding or adding-deleting a black corner) or adding a black corner to the current voxel.

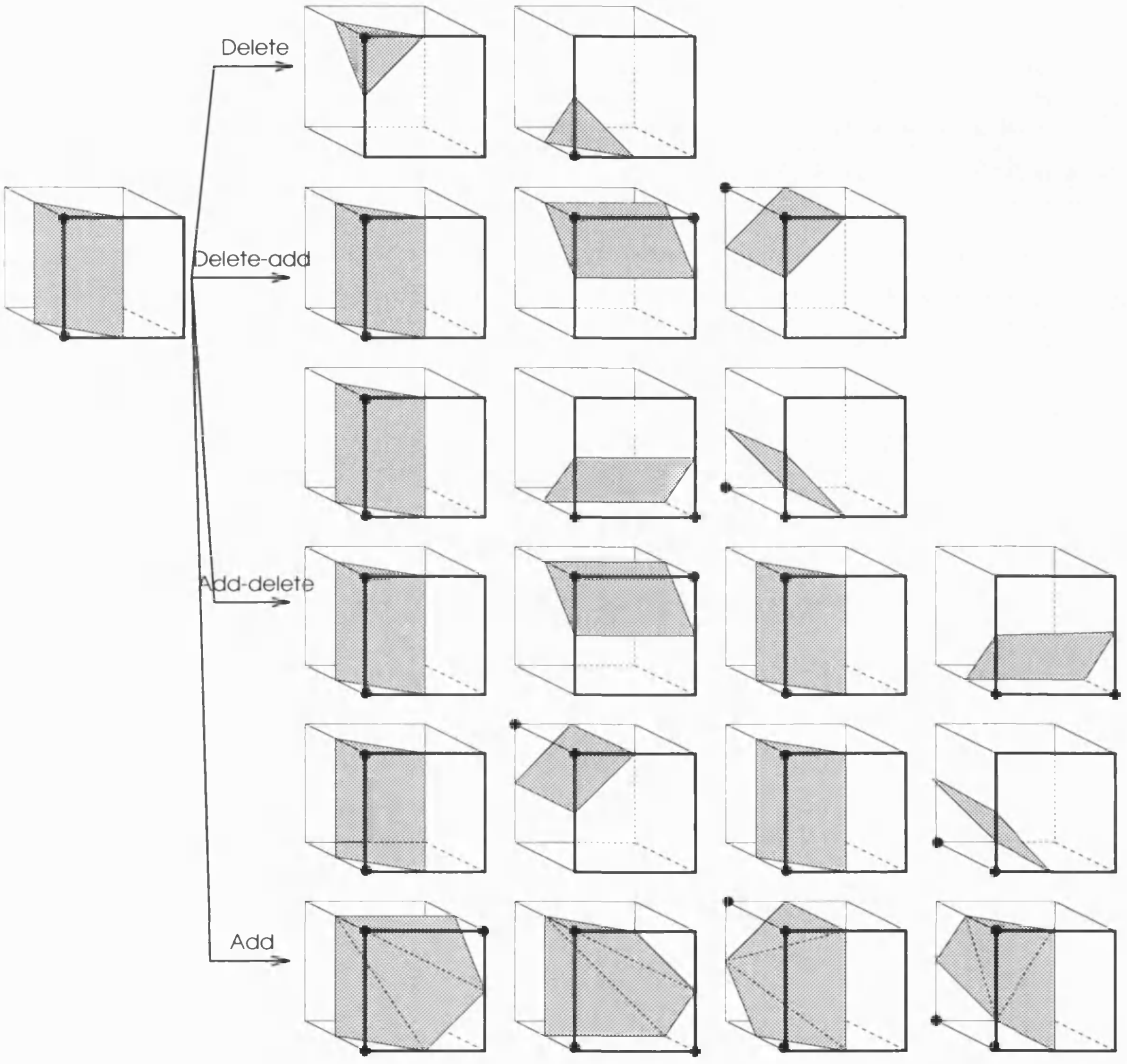

Figure A-3: PROPOSALS WHEN MOVING FROM A 2–CORNER VOXEL. This figure is similar to Figure A-2 but shows the potential updates for voxels with *two* black corners.

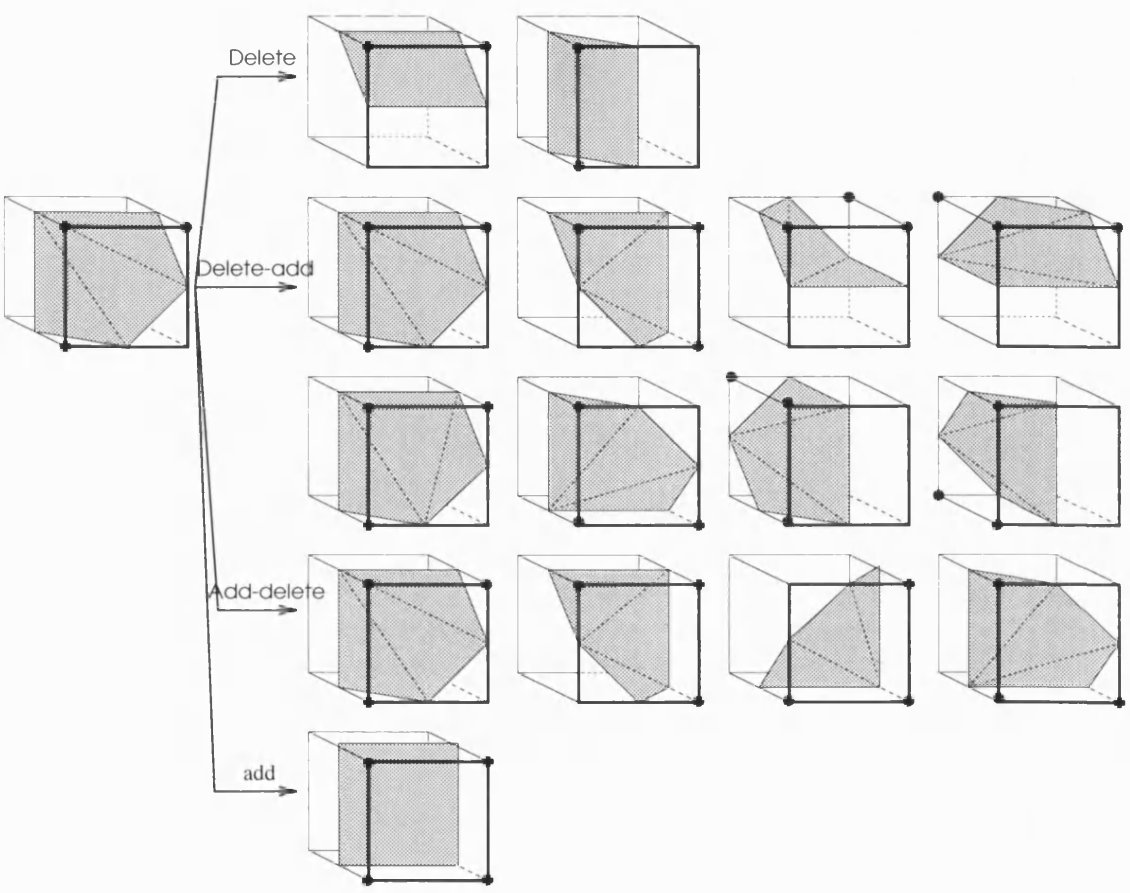

Figure A-4: PROPOSALS WHEN MOVING FROM A 3-CORNER VOXEL. This figure is similar to Figure A-2 but shows the potential updates for voxels with *three* black corners.

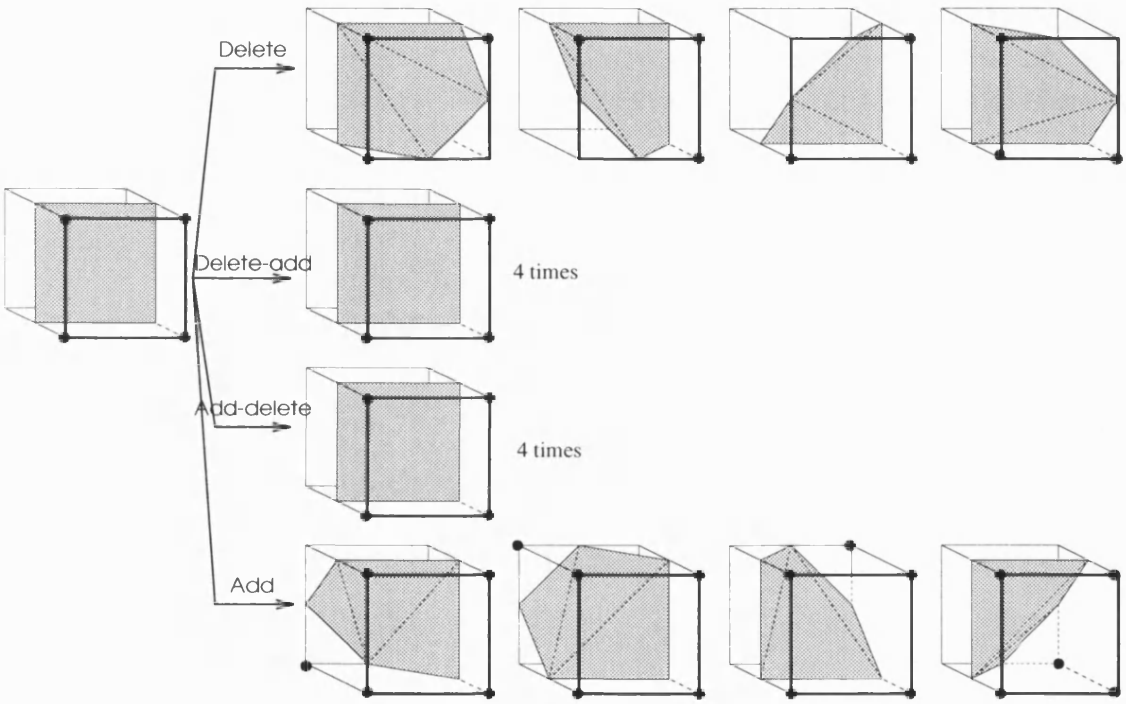

Figure A-5: PROPOSALS WHEN MOVING FROM A 4–CORNER VOXEL. This figure is similar to Figure A-2 but shows the potential updates for voxels with *four* black corners.

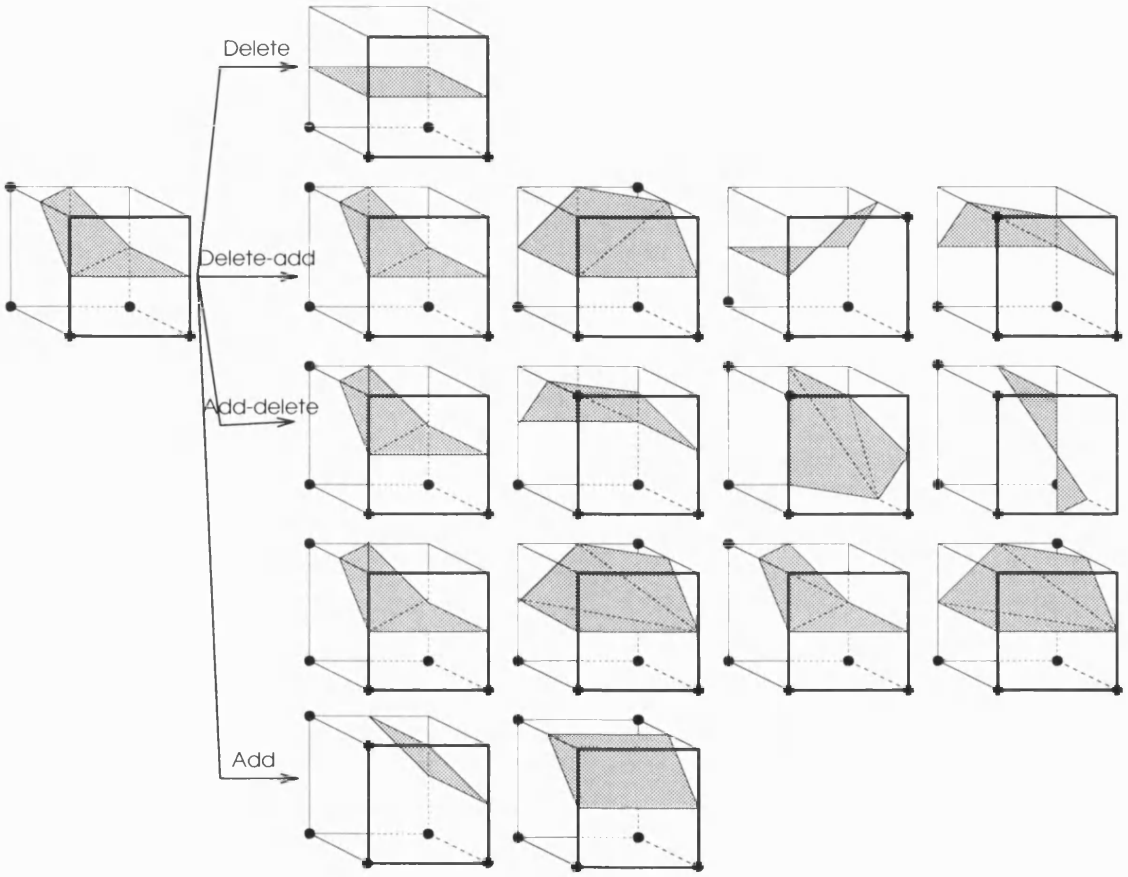

Figure A-6: PROPOSALS WHEN MOVING FROM A 5–CORNER VOXEL. This figure is similar to Figure A-2 but shows the potential updates for voxels with *five* black corners. Compare this set of potential updates for a 5–corner voxel with the list of proposals in Figure 5-14, where all 5–corner voxels are considered.

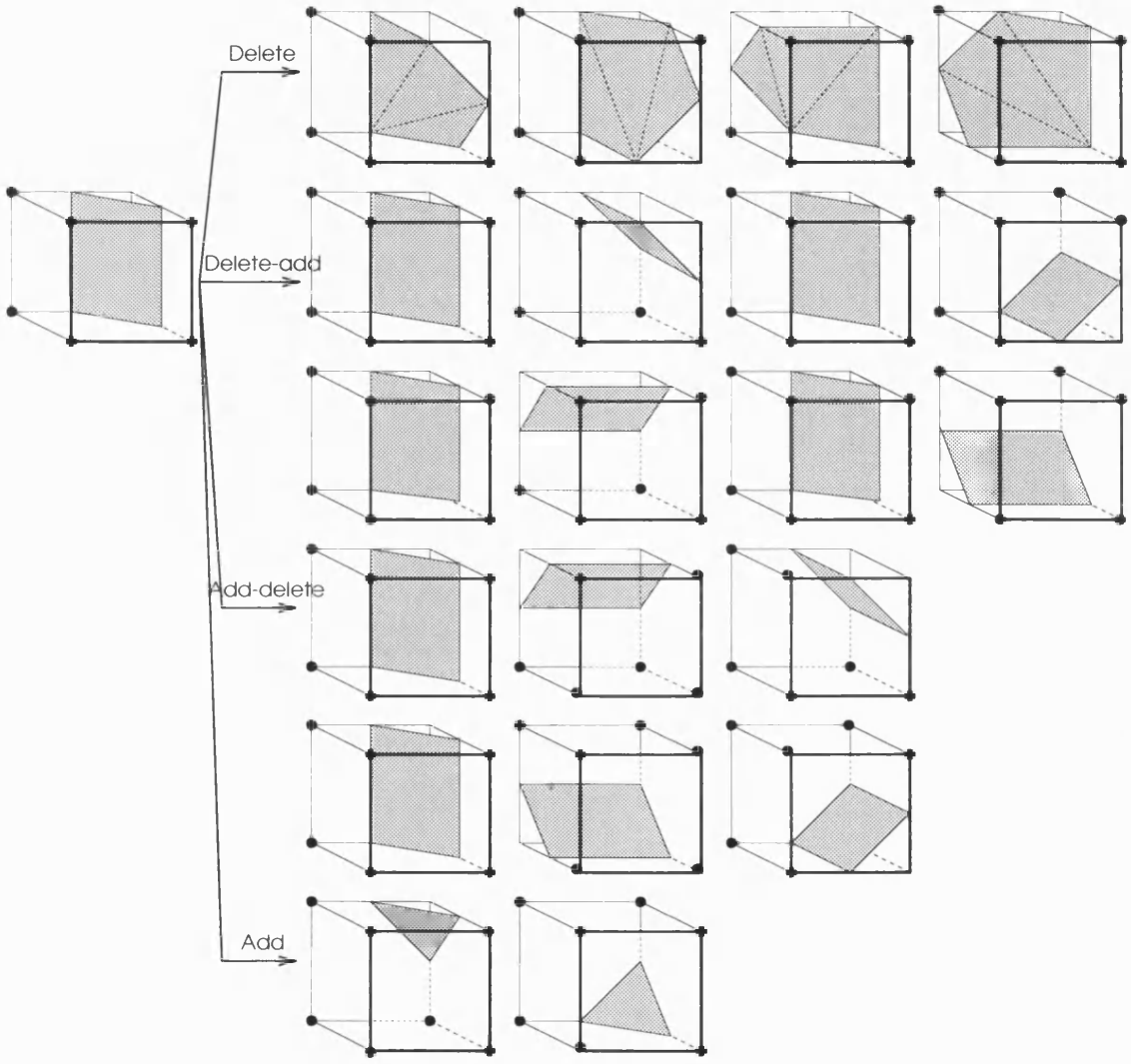

Figure A-7: PROPOSALS WHEN MOVING FROM A 6–CORNER VOXEL. This figure is similar to Figure A-2 but shows the potential updates for voxels with *six* black corners.

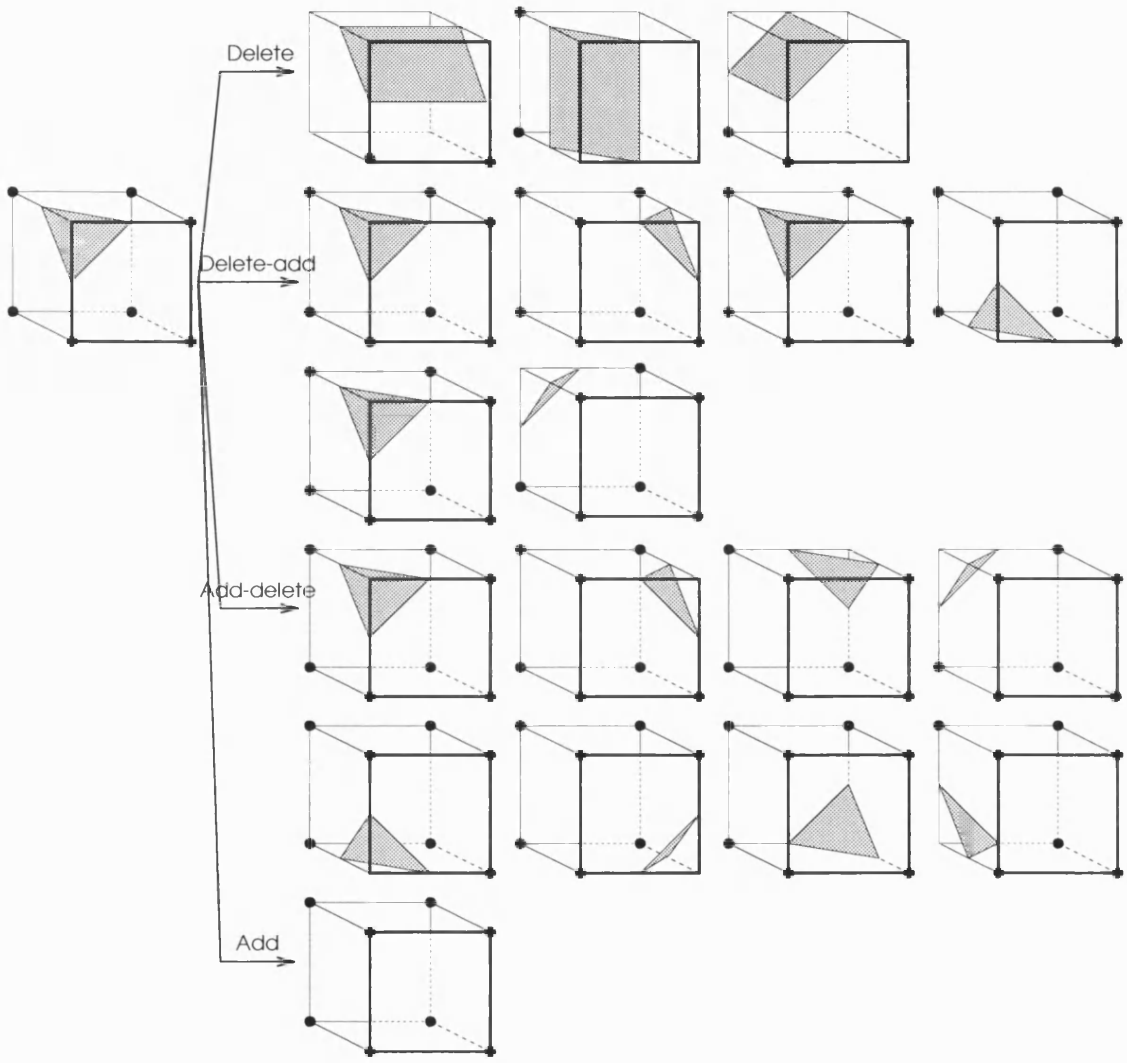

Figure A-8: PROPOSALS WHEN MOVING FROM A 7–CORNER VOXEL. This figure is similar to Figure A-2 but shows the potential updates for voxels with *seven* black corners.

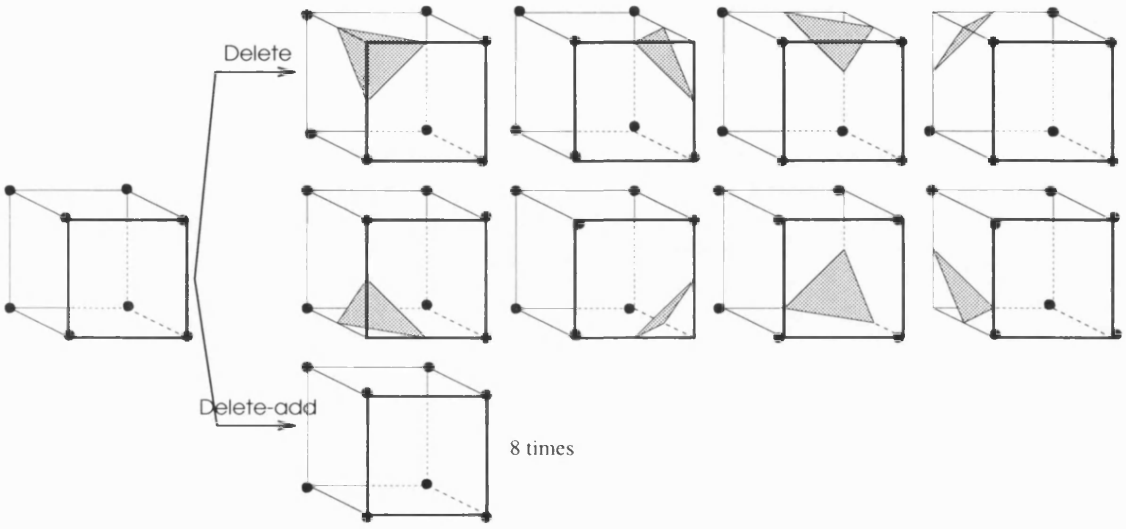

Figure A-9: PROPOSALS WHEN MOVING FROM A 8–CORNER VOXEL. This figure is similar to Figure A-1 but shows the potential updates for voxels with *eight* black corners.

Appendix B

Computational issues

It is unworthy of excellent men to lose hours
like slaves in the labour of calculation
which could safely be relegated to anyone else
if machines were used.
Gottfried Von Leibniz

There are two ways of constructing
a software design. One way is to make it so
simple that there are obviously no deficiencies.
And the other way is to make it so complicated
that there are no obvious deficiencies.
C. A. R. Hoare

This appendix briefly mentions some of the computational issues that arose while completing this thesis. Although they are not central to the issues raised in the body of the thesis, similar problems are likely to be encountered by others working in statistical image analysis, especially when writing software.

The issues discussed are:

- the use of object-oriented languages for combinatorial optimisation,
- numerical instability in the simulated annealing algorithm,
- the mechanisms used to visualise 3D data
- the choice of random-number generator.

B.1 Languages and data structures

One of the major differences between combinatorial computing and other areas of computing such as statistics, numerical analysis and linear programming is the use of complex data types. While built in data types such as integers, reals, vectors and matrices are usually sufficient in other areas, combinatorial computing relies heavily on types like dictionaries, sequences, graphs, points, segments, *etc.*

The vast majority of the work for this thesis consists of writing software to define objects like images, pixels, voxels, *etc.* This software relies on object-oriented programming (OOP).

B.1.1 Why use an OOP language for image analysis?

Most of the algorithms discussed in this thesis are implemented using an object-oriented language C++. In the past, procedure oriented languages, such as `C` and `Fortran`, involved learning how to call a library of perhaps several hundred functions. It was essential to learn how to call and use each function separately. For example, one function might call another which then updates a data structure which may lie in another file. This approach quickly becomes unmanageable. C++ offers two features to address these problems: data abstraction and OOP.

Starting from C, which offers flexibility and efficiency, and adding data abstraction plus OOP results in C++. Data abstraction means packaging the data structures and functions which manipulate these data structures, into a user-defined data type that is easy to use. We also want new data types to be efficient and do type checking *etc.* OOP offers the user the opportunity to reuse code, by expressing any commonality among data types as a hierarchy of different but related types or classes. The types at the top of this tree are very general and become more refined as you progress down the tree.

B.1.2 Basic building blocks for image analysis

To avoid the time consuming task of having to build up a library of objects from scratch, a C++ library of efficient data types and algorithms for combinatorial computing, called `LEDA` (Naeher 1995), is used to build many of the data structures needed. `LEDA` provides a sizable collection of data types and algorithms in a form which allows them to be used by non-computer scientists.

Example B.1. THE GRAPH DATA STRUCTURE IN LEDA .
`LEDA` contains an elaborate data type for creating graphs, where a `graph` G consists of a list of nodes V and a list of edges E, with a pair of `nodes` $(v, w) \in V \times V$ associated with every `edge` $e \in E$. `LEDA` offers the standard iterations over G, such as 'for all nodes v in the `graph` G do' or 'for all neighbours w of the node v do'. Addition and deletion of vertices and edges is possible. It also offers arrays and matrices indexed by nodes and edges. The ideal is to be able to use the data type `graph` to design software for solving `graph` problems in a form that is close to the typical text book presentation. □

See page 142 for details of the other languages used in this thesis.

B.2 Numerical instability in algorithms

This section summarises the effect of numerical precision on the simulated annealing algorithm. In particular, an example is given to show how easy it is for instability to arise in a binary image. A possible solution is outlined.

When running simulated annealing, the objective is to minimise an energy function, E. Part of the algorithm involves calculating the energy for a proposal, dividing by the temperature and then exponentiating the result. If the noise is low, $\sigma^2 < 0.001$, or the

temperature is near zero, the exponentiation can lead to very small numbers. Subsequently normalising this set of numbers, one for each proposal, can lead to division by zero errors.

Example B.2. NUMERICAL INSTABILITY.
At any Gibbs update, the energy for all possible moves to a new state, i, are calculated, $\{U_i : i = 1,\ldots,n\}$, where $n = 2^1$ for a single pixel update or $n = 2^4$ when updating a 2×2 sub-window simultaneously. The kernel of the annealing algorithm is the decision to accept a proposal to move to a new state which has higher energy than the current state. The probability of accepting state i as the new state is

$$\Pr(X = i) = \exp\{-U_i/T\} / \sum_{j=1}^{n} \exp\{-U_j/T\} \tag{B.1}$$

where T is the temperature. As the temperature drops to zero the algorithm behaves like the ICM algorithm of Besag (1986), at least in theory.

However, in the single precision (32 bit) IEEE standard, the smallest non-zero random floating point number that is available is $2^{-126} \approx 1.175e^{-38}$ and for the double precision (64 bit) IEEE standard the corresponding number is $2^{-1022} \approx 2.225e^{-308}$. These limitations can be exceeded quite easily during a typical simulation.

Suppose we observe the following 3×2 image $\left(\begin{smallmatrix} 0.988 & 0.999 \\ 1.005 & -0.010 \\ 0.018 & 0.994 \end{smallmatrix}\right)$, where the variance of the signal is known to be $\sigma^2 = 0.0001$. The closest mean classifier is $\left(\begin{smallmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$. Updating one pixel at a time using a first order neighbourhood model with $\beta = 4$ means that for the pixel at row two, column two, say, the choice is between 0 and 1 with corresponding energy values of

$$U_0 = \big(0 - (-0.010502)\big)^2/(2 * 0.0001) + 4 * 3 = 12.55146$$

and

$$U_1 = \big(1 - (-0.010502)\big)^2/(2 * 0.0001) + 4 * 0 = 5105.57. \tag{B.2}$$

with probabilities

$$\begin{aligned} \Pr(X = 0) &= \exp\{-U_0/T\} / \sum_{j=1}^{2} \exp\{-U_j/T\} \\ &= \exp\{-12.55/T\}/(\exp\{-12.55/T\} + \exp\{-5105.57/T\}) \end{aligned}$$

and

$$\begin{aligned} \Pr(X = 1) &= \exp\{-U_1/T\} / \sum_{j=1}^{2} \exp\{-U_j/T\} \\ &= \exp\{-5105.57/T\}/(\exp\{-12.55/T\} + \exp\{-5105.57/T\}). \end{aligned}$$

Now suppose our temperature is at $T = 0.1$ which appears to be a long way from the case $T \approx 0$, where we might expect instability to occur. In this case, $\Pr(X = 0) \propto \exp\{-125.5\} \approx 3.134e^{-55}$ which is zero in the single-precision, IEEE standard and clearly $\Pr(X = 1) \propto \exp\{-51005.7\}$ is also zero. So *both* choices generate a '0/0' probability. With double precision, $\Pr(X = 0)$ can be safely calculated and the algorithm would choose $X = 0$ as the update without any modification being necessary. However, even for a larger variance the decrease in temperature towards zero will eventually cause instability and lead to a probability of '0/0' for all proposals. Stiles (1993) offers further details about the limits of various machines. □

B.2.1 A remedy for numerical instability

As only the relative differences between the energy values is relevant, the problem is best resolved by first subtracting the minimum energy from the energy value for each of the proposals (Gavin 1993). This guarantees that at least one proposal has a reasonable probability of being accepted.

The energy for each proposal is first calculated. The minimum energy is then subtracted from all the energies. This doesn't affect the relative energy values and hence the probabilities are unaffected. Now the probability for the smallest energy is never '0/0', so at least one proposal is reasonable. For high energy proposals the probability is not calculable but is approximated by zero which is acceptable in that instance. In particular, if all of the proposals have high energy values then it is the proposal with the minimum energy that is chosen. This is equivalent to an ICM (Besag 1986) update for that *single* pixel. Other pixels in the same sweep may be unaffected, so the energy for the scene as a whole may still rise.

B.3 Storing 3D reconstructions

It is informative to consider briefly how the reconstructions in Chapter 5 are stored and drawn. The purpose is to highlight some of the problems associated with 3D visualisation.

The reconstructions from the algorithm in Chapter 5 are written in ASCII format. The '#' symbol denotes a comment, so everything on the line beyond that point is ignored. Each reconstruction is formatted as a list (`LIST`) of voxels (`OFF` objects). There is one `OFF` object for each voxel that has some black in it. To save space, only information about the black in a voxel is recorded. Each voxel is represented by a list of 3D points, one for each black vertex in the voxel. (The points are absolute positions in 3D, real-world space, as opposed to relative, model-world values.) The connectivity between vertices results in a list of faces for each voxel. This is represented by a list of integers, one for each face. Each integer is the subscript of the 3D point on the list of vertices. The first integer on each list specifies the number of indices, hence vertices, on that face. The number of vertices and faces in each voxel is also needed to draw each voxel but this information is easily deduced from the other details.

```
{LIST # List of voxels for each image.

{OFF # Cube for vertex(2,3,4)
10 9 # number of vertices and faces
# vertices: bottom-left-front to top-right-back vertex
2 3 4
3 3 4
2 6 4
3 6 4
2 3 8
4 3 8
4 3 6
4 4.5 8
2 6 8
3 6 8
# name the faces
4 2 3 1 0 # 1front face.
5 8 9 7 5 4 # 2back face.
4 4 8 2 0 # 3left face.
3 7 6 5 # 4right face.
4 8 9 3 2 # 5top face.
5 4 5 6 1 0 # 6bottom face.
3 1 7 9 # 7inner1 face.
3 1 6 7 # 8inner2 face.
3 1 3 9 # 9inner3 face.
} # end of vertex    2.00    3.00    4.00

# Vertex (2,3,8) is ALL white

{OFF # Cube for vertex(6,8,4)
8 8 # number of vertices and faces
# vertices: bottom-left-front to top-right-back vertex
8 8 4
7 8 4
...
} # end of vertex    6.00    8.00    4.00
} # end of LIST
```

Figure B-1: EXTRACT FROM THE OUTPUT FILE FOR A 3D OBJECT. Each reconstruction is a list (`LIST`) of voxels (`OFF` objects). Only information about the black in each voxel is recorded to save space. The black region within each voxel is represented by a list of 3D points, one for each black vertex in the voxel. The connectivity between vertices results in a list of faces for each voxel. This is represented by a list of integers, one for each face. Each integer is the subscript of the 3D point on the list of vertices. The first integer on each of these lists specifies the number of indices, hence vertices, on that face. Comment lines follow the hash symbol '#'.

For example, the first voxel in Figure B-1 has ten vertices and nine faces. The ten 3D vertices are listed, one on each line. Then for each of the nine faces, the subscripts for the vertices on that face are listed. (The n subscripts on a face are numbered from 0 to $n-1$.) The first number on each list is just the number of vertices on that face of the voxel. For example, the 'right' face in the voxel is a triangle formed from the three subscripts $\{7,6,5\}$ in the list of vertices. These points are $\{(4,4.5,8),(4,3,6),(4,3,8)\}$, respectively.

The display engine `Geomview` is used to draw the 3D images shown in this thesis (see page 142).

B.4 Random number generators

A variety of random number generators are used for the simulations shown in this thesis. The principle generator due to Wichmann and Hill (1982) combines three multiplicative generators. Although this makes the generator expensive to call, it also results in a huge period. De Matteis and Pagnutti (1993) claim that this generator compares favourably with others.

Bibliography

AMS (1995, January). *$\mathcal{A}_{\mathcal{M}}$S-LaTeX Version 1.2*. Providence, RI: American Mathematical Society.

Atiquzzaman, M. (1992). Multiresolution Hough transform – an efficient method of detecting patterns in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*(11), 1090–1095.

Besag, J. E. (1974). Spatial interaction and the statistical analysis of lattice systems (with discussion). *Journal Royal Statistical Society, Series B 36*, 192–236.

Besag, J. E. (1986). On the statistical analysis of dirty pictures (with discussion). *Journal Royal Statistical Society, Series B 48*, 259–302.

Besag, J. E. (1989). Towards Bayesian image analysis. *Journal of Applied Statistics 16*(3), 395–407.

Besag, J. E. and P. J. Green (1992). Spatial statistics and Bayesian computation (with discussion). *Journal Royal Statistical Society, Series B 55*, 25–37.

Besag, J. E. and P. J. Green (1993). Markov chain Monte Carlo in statistical physics and Bayesian statistics. Mathematics Research Report S-93-04, University of Bristol.

Besag, J. E., P. J. Green, D. Higdon, and K. Mengersen (1995). Bayesian computation and stochastic systems. *Statistical Science 10*(1), 3–66.

Bhattacharya, A., C. S. S. Reddy, and S. K. Srivastav (1993). Remote sensing for active volcano monitoring in Barren Island, India. *Photogrammetric Engineering and Remote Sensing LIX*(8), 1293.

Boult, T. E. and G. Wolberg (1993). Local image reconstruction and subpixel restoration algorithms. *CVGIP: Graphical Models and Image Processing 55*(1), 63–77.

Chen, H., J. R. Swedlow, M. G. J. W. Sedat, and D. A. Agard (1995). The collection, processing, and display of digital three-dimensional images of biological specimens. In J. B. Pawley (Ed.), *Handbook of Biological Confocal Microscopy* (Second ed.)., Chapter 13, pp. 197–210. New York: Plenum Press.

Cline, H. E., W. E. Lorensen, S. Ludke, C. R. Crawford, and B. C. Teeter (1988). Two algorithms for the three-dimensional reconstruction of tomograms. *Medical Physics 15*(3), 320–327.

Crawford, J. W., K. Ritz, and I. M. Young (1993). Quantification of fungal morphology, gaseous transport and microbial dynamics in soil: An integrated framework using fractal geometry. *Geoderma 56*, 157–172.

De Matteis, A. and S. Pagnutti (1993). Long-range correlation analysis of the Wichmann-Hill random number generator. *Statistics and Computing* (3), 67–70.

Feller, W. (1968). *An Introduction to Probability and Its Applications* (Third Edition ed.), Volume 1. New York: John Wiley.

Foody, G. M. (1994). Ordinal-level classification of sub-pixel tropical forest cover. *Photogrammetric Engineering and Remote Sensing LX*(1), 61.

Fricker, M. D. and N. S. White (1992). Application of confocal microscopy and three-dimensional image analysis to plant and microbial cells. *BINARY 4*, 44–49.

Friedland, N. and D. Adam (1989). Ventricular cavity boundary detection from sequential ultrasound images using simulated annealing. *IEEE Transactions on Bio-Medical Engineering 8*(4).

Gatherer, A. and T. Meng (1991). A lower bound on alignment accuracy and subpixel resolution in lithography. *Journal of Vacuum Science & Technology B - Microelectronics Processing and Phenomena 9*(6), 3601–3605.

Gatherer, A. and T. Meng (1992). Robust subpixel alignment in lithography. *Journal of Vacuum Science & Technology B 10*(6), 2662–2666.

Gavin, J. and C. Jennison (1995). A subpixel reconstruction algorithm. *Proc. of the Statistical Computing Section, American Statistical Association.*

Gavin, J. B. (1993). A note on the effect of numerical instability upon the simulated annealing algorithm. Technical report. University of Bath.

Geman, D., S. Geman, and C. Graffigne (1987). Locating texture and object boundaries. In P. A. Devijver and J. Kittler (Eds.), *Pattern Recognition Theory and Applications*, Volume F30, pp. 165–177. Berlin: Springer-Verlag.

Geman, D., S. Geman, C. Graffigne, and P. Dong (1990). Boundary detection by constrained optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12*, 609–628.

Geman, S. and D. Geman (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 6*, 721–741.

Geyer, C. J. (1993). Practical Markov chain Monte Carlo (with discussion). *Statistical Science 8*, 473–483.

Gidas, B. (1989). A renormalization group approach to image processing problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11*, 164–180.

Glasbey, C. A. (1994). Inference on binary images from binary data. To appear in *Advances in Applied Probability*, 1–15.

Glasbey, C. A. and G. W. Horgan (1994). *Image Analysis for the Biological Sciences.* Chichester, England: John-Wiley & Sons.

Green, P. J. (1994a). Contribution to the discussion of the paper by Grenander and Miller. *Journal Royal Statistical Society, Series B 56*, 589–590.

Green, P. J. (1994b). Highly Structured Stochastic Systems. Technical report, University of Bristol. An Introduction to the European Science Foundation scientific network on Highly Structured Stochastic Systems.

Green, P. J. (1995a). Markov chain Monte Carlo in image analysis. In Gilks, Richardson, and Spiegelhalter (Eds.), *Practical Markov chain Monte Carlo*, pp. 1–20. Chapman and Hall.

Green, P. J. (1995b). Monte Carlo methods: An overview. In D. M. Titterington (Ed.), *Proceedings of the IMA Conference on Complex Stochastic Systems and Engineering*, pp. 183–190. IMA Conference Series No. 54.

Green, P. J. (1995c). Reversible jump MCMC computation and Bayesian model determination. *Biometrika.*

Greig, D., B. Porteous, and A. Scheult (1989). Exact maximum a posteriori estimation for binary images. *Journal Royal Statistical Society, Series B 51*, 271–279.

Grenander, U. (1983). Tutorial in pattern theory. Technical report, Division of Applied Mathematics, Brown University, Providence.

Grenander, U. and M. Miller (1994). Representations of knowledge in complex systems (with discussion). *Journal Royal Statistical Society, Series B 56*, 549–603.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika 57*(1), 97–109.

Helterbrand, J. D., N. Cressie, and J. L. Davidson (1994). A statistical approach to identifying closed object boundaries in images. *Advances in Applied Probability 26*, 831–854.

Hitchcock, D. and C. A. Glasbey (1994). Binary image restoration at subpixel resolution from multi-level data. Technical report, Scottish Agricultural Statistics Service, JCMB, King's Buildings, Edinburgh, EH9 3JZ, Scotland.

Howard, V. (1990). *The Confocal Microscope as an Instrument for Measuring Microstructural Geometry.* Academic Press Ltd.

Huertas, A. and G. Medioni (1986). Detection of intensity changes with subpixel accuracy using Laplacian-Gaussian masks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, 651–663.

Hurn, M. A. (1992). *Models and Algorithms for Image Reconstruction.* PhD thesis, University of Bath.

Hurn, M. A. and C. Jennison (1995). A study of simulated annealing and a revised cascade algorithm for image reconstruction. *Statistics and Computing 5*, 175–190.

Jasinski, M. F. (1990a). Functional relation among subpixel canopy cover, ground shadow, and illuminated ground at large sampling scales. *Proceedings of the Society of Photo-optical Instrumentation Engineers 1300*(20), 48–58.

Jasinski, M. F. (1990b). Sensitivity of the normalized difference vegetation index to subpixel canopy cover, soil albedo, and pixel scale. *Remote Sensing of Environment 32*(2–3), 169–187.

Jasinski, M. F. and P. S. Eagleson (1990). Estimation of subpixel vegetation cover using red infrared scattergrams. *IEEE Transactions on Geoscience and Remote Sensing 28*(2), 253–267.

Jennison, C. and M. Jubb (1988). Statistical image restoration and refinement. In C. N. de Graaf and M. A. Viergerer (Eds.), *Information Processing in Medical Imaging*, pp. 255–262. New York: Plenum Press.

Jubb, M. and C. Jennison (1991). Aggregation and refinement in binary image restoration. In A. Possolo (Ed.), *Spatial Statistics and Imaging*, Volume 20 of *Lecture Notes*, pp. 150–162. Hayward, California, USA: IMS.

Kass, M., A. Witkin, and D. Terzopoulos (1987). Snakes: Active contour models. *International Journal of Computer Vision*, 321–331.

Kent, J. T. and K. V. Mardia (1988). Spatial classification using fuzzy membership models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 10*, 661–671.

Kirkpatrick, S., C. D. Gelatt, Jr., and M. P. Vecchi (1982). Optimization by simulated annealing.

Kiryati, N. and A. Bruckstein (1991). Gray levels can improve the performance of binary image digitizers. *CVGIP: Graphical Models and Image Processing 53*(1), 31–39.

Koplowitz, J. and A. M. Bruckstein (1989). Design of perimeter estimators for digitized planar shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11*, 611–622.

Koplowitz, J. and A. P. Sundar Raj (1987). A robust filtering algorithm for subpixel reconstruction of chain coded line drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence 9*, 451–457.

Lamport, L. (1986). *LaTeX: A Document Preparation System.* Reading, Ma.: Addison-Wesley Publishing Co.

Levine, M. D. (1985). *Vision in Man and Machine.* McGraw-Hill Series in Electrical Engineering. New York: McGraw-Hill.

Lin, W. E. and H. E. Cline (1989). A new surface interpolation technique for recconstruction of 3D objects from serial cross-sections. *Computer Vision, Graphics and Image Processing 48*, 124–143.

Lorensen, W. E. and H. E. Cline (1987). Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics 21*(4), 163–169.

Lyvers, E. P., O. R. Mitchell, M. L. Akey, and A. P. Reeves (1989). Subpixel measurements using a moment-based edge operator. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11*, 1293–1309.

Marr, D. and E. Hildreth (1980). Theory of edge detection. *Proc. R. Soc. Lond. 207*, 187–217.

Marroquin, J., S. Mitter, and T. Poggio (1987). Probabilistic solution of ill-posed problems in computational vision. *Jour. Amer. Statist. Assoc. 82*, 76–89.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953). Equations of state calculations by fast computing machines. *J. of Chemical Physics 21*, 1087–1092.

Miller, J. V., D. E. Breen, W. E. Lorensen, R. M. O'Bara, and M. J. Wozny (1991). Geometrically deformed models: A method for extracting closed geometric models from volume data. *Computer Graphics 25*(4), 217–226.

Mort, M. S. and M. D. Srinath (1988). Maximum likelihood image registration with subpixel accuracy. *Proc. of Soc. of Photo-Optical Instrumentation Engineers 1974*, 38–45.

Naeher, S. (1995). *LEDA: A Library of Efficient Data Types and Algorithms* (V3.1.1 ed.). Im Stadtwald, 6600 Saarbruecken, F.R.G.: Max-Planck-Institut fuer Informatik. Email: leda@mpi-sb.mpg.de.

Oakley, J. P. and R. T. Shann (1991). Efficient method for finding the position of object boundaries to sub-pixel precision. *Image and Vision Computing 9*(4), 262–272.

O'Sullivan, F. and M. Qian (1994). A regularized contrast statistic for object boundary estimation–implementation and statistical evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*(6), 561.

Phillips, D. B. and A. F. M. Smith (1994). Bayesian model comparision via jump diffusions. Research report, Imperial College of Science, Technology and Medicine.

Ripley, B. D. (1987). *Stochastic Simulation.* New York: John Wiley and Sons.

Ripley, B. D. (1988). *Statistical Inference for Spatial Processes.* Cambridge, U.K.: Cambridge University Press.

Ritz, K. and J. W. Crawford (1990). Quantification of the fractal nature of colonies of *Trichoderma viride. Mycological Research 94*, 1138–1141.

Roberts, G. O. and N. G. Polson (1994). On the geometric convergence of the Gibbs sampler. *Journal Royal Statistical Society, Series B 56*, 377–384.

Roberts, L. G. (1965). Machine perception of three-dimensional solids. In J. T. T. *et al.* (Ed.), *Optical and Electro-Optical Information Processing.* Cambridge, MA: M.I.T. Press.

Rosenfeld, A. (1992). Survey: Image analysis and computer vision – 1991. *CVGIP: Image Understanding 55*(3), 349–380.

Rosenfeld, A. (1993). Survey: Image analysis and computer vision – 1992. *CVGIP: Image Understanding 58*(1), 85–135.

Rosenfeld, A. and A. C. Kak (1982). *Digital Picture Processing* (Second ed.). New York, NY, U.S.A.: Academic Press.

Rosenthal, J. S. (1994). Theoretical rates of convergence for Markov chain Monte Carlo. In *Proceedings of Interface '94.*

Sandor, T. and J. R. Spears (1985). Statistical considerations on the precision of assessing blood vessel diameter in cine coronary angiography. *Computers and Biomedical Research 18*(6), 531–543.

Scott, D. W. (1992). *Multivariate Density Estimation; Theory, Practice and Visualisation.* John Wiley & Sons.

Serra, J. (1982). *Image Analysis and Mathematical Morphology.* London: Academic Press.

Shaw, P. J. and D. J. Rawlins (1991). Three-dimensional florescence microscopy. *Prog. Biophys. Molec. Biol. 56*, 187–213.

Silverman, B. W. and C. Jennison (1991). How to charge for boundaries in a pixel image. In K. V. Mardia (Ed.), *The Art of Statistical Science: A Tribute to G. S. Watson*, pp. 209–230. New York: Wiley & Sons.

Silverman, B. W., C. Jennison, J. Stander, and T. C. Brown (1990). The specification of edge penalties for regular and irregular pixel images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12*, 1017–1024.

Skilling, J. and R. K. Bryan (1984). Maximum entropy image reconstruction: General algorithm. *Monthly Notices of the Royal Astronomical Society 211*, 111–124.

Smith, A. F. M. and A. E. Gelfand (1992). Bayesian statistics without tears: a sampling-resampling approach. *The American Statistician 46*(2), 84–88.

Sokal, A. D. (1989). Monte Carlo methods in statistical mechanics: Foundations and new algorithms. In *Cours de Troisième Cycle de la Physique en Suisse Romande.* Lausanne.

Sriraman, R., J. Koplowitz, and S. Mohan (1989). Tree searched chain coding for subpixel reconstruction of planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11*, 95–104.

Srivastava, A., M. I. Miller, and U. Grenander (1991). Jump diffusion processes for object tracking and direction tracking and direction finding. In *Proc. 29^{th} Annual Allerton Conference on Communication, Control and Communication*, pp. 563–570.

Stander, J. and B. W. Silverman (1994). Temperature schedules for simulated annealing. *Statistics and Computing 4*(1), 21–32.

Stiles, G. S. (1993). The effect of numerical precision upon simulated annealing. Technical report, Dept. of Elec. Eng., Utah State Uni., Logan, Utah.

Storvik, G. (1994). A Bayesian approach to dynamic contours through stochastic sampling and simulated annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*(10), 976.

Swendsen, R. and J. Wang (1987). Nonuniversal critical dynamics in Monte Carlo simulations. *Physical Review Letters 58*, 86–88.

Szirányi, T. (1992). Statistical subpixel measurement technology of light-beam width and profiles. *Optics and Lasers in Engineering 16*, 1–15.

Szirányi, T. (1993). Noise effects in statistical subpixel pattern recognition. *Lecture notes in Computer Science 719*, 74–81.

Tabatabai, A. J. and O. R. Mitchell (1984). Edge location to subpixel values in digital imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence 6*, 188–201.

Tian, Q. and M. N. Huhns (1986). Algorithms for subpixel registration. *Computer Vision, Graphics and Image Processing 35*, 220–233.

Weir, N. and S. Djorgovski (1991). A subpixel deconvolution method for astronomical images. *Fundamental Theories of Physics 43*(37), 275–283.

West, G. A. W. and T. A. Clarke (1990). A survey and examination of subpixel measurement techniques. *Proceedings of the Society of Photo-optical Instrumentation Engineers 1395*, 456–463.

Wichmann, B. A. and I. D. Hill (1982). Algorithm AS 183 — an efficient and portable psuedo-random number generator. *Applied Statistics 31*, 188–190.

Wolberg, G. and T. Pavlidis (1985). Restoration of binary images using stochastic relaxation with annealing. *Pattern Recognition Letters 3*, 375–388.

Young, R. A. (1987). Locating industrial parts with subpixel accuracies. *Proceedings of the Society of Photo-optical Instrumentation Engineers 728*(33), 2–9.

Index

Mol a dheireadh

(Praise the end of it)